AN EVOLUTIONARY METHOD FOR SYNTHESIZING TECHNOLOGICAL PLANNING AND ARCHITECTURAL ADVANCE

A Thesis
Presented to
The Academic Faculty

by

Bjorn Forstrom Cole

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Daniel Guggenheim School of Aerospace Engineering

Georgia Institute of Technology
May 2009

AN EVOLUTIONARY METHOD FOR SYNTHESIZING TECHNOLOGICAL PLANNING AND ARCHITECTURAL ADVANCE

Approved by:

Professor Dimitri N. Mavris, Advisor
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Carlee A. Bishop
Georgia Tech Research Institute

Professor Brian J. German
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Frederic Villeneuve
Siemens

Professor Mark Costello
School of Aerospace Engineering
Georgia Institute of Technology

Date Approved: May 13, 2009

To a future worthy of the children of the stars.

ACKNOWLEDGEMENTS

This thesis has been under development for quite some time, and so there are a few people to thank for helping me at different times.

I would like to thank my advisor, Dr. Dimitri Mavris, for doing all he can to provide a stable environment in which to conduct academic research with the proper care and independence. This brings him a great deal more stress in his daily life and longer hours than if he had merely set out to secure for himself a professorship. I will also likely owe him a debt for insisting that I keep my eyes open to often less-than-pleasant and non-academic realities of the modern world of organized research.

I would also like to thank my reading committee members Dr. Brian German, and Dr. Carlee Bishop, who have guided my work and ways in which to demonstrate its relevance to the design community. Both have been extremely thoughtful in their comments and kind in nature. This is also true for Dr. Mark Costello, whose counsel I must admit to having sought too little of, but was given in a generous spirit when asked.

In the formative period of my thesis, I spoke to many colleagues about potential avenues to pursue. For their help, I thank Dr. Holger Pfaender, Dr. Rob McDonald, Dr. Jan Osburg, Dr. Taeyun Choi, Dr. Stephane Dufresne and Dr. Frederic Villeneuve. To Dr. Villeneuve, I owe a double debt of inspiration and willingness to guide my work as a member of my committee.

As for aid on this thesis as currently formulated, I have a few more people to thank. Santiago Balestrini-Robinson, William Engler, Kelly Griendling, Daniel Cooksey, Dr. Diana Talley, Carl Johnson and Shuo-Ju Chou have been helpful in guiding me through this process. Santiago receives special credit for nearly capsizing my research and then strengthening it through a direction of reading I had not considered, namely the development of multidimensional growth models. In addition to feedback on areas in

which they were better read than I, they were also invaluable colleagues with whom to discuss general ideas and to exchange ideas as to what the research process is all about. They have also patiently reviewed the rougher versions of my proposal presentation.

Throughout the period between my proposal and my defense, I have been very fortunate in having a large number of colleagues to observe move through this process. They have prompted me to attempt to improve in every area, from writing and presenting to attempting to maintain rigor in an area of research that, after all, has no laboratory bench experiments. I would like to think Cyril de Tenorio and Mike Armstrong for exchanging ideas, as their dissertation work has had many inadvertent commonalities with my own, such as a desire for a flexible way to combine elements into a system. They also brought to my attentions concepts such as induced functions and potential ways to perform analysis of an architecture at multiple operational conditions.

I have had the good fortune to have many friends on both sides of the "wall" between student life and life as a doctoral researcher. For their insights on the world of professional research, I would like to thank Dr. Simon Briceno, Dr. Kyle Collins, Dr. Alex Moodie, Dr. Scott Duncan, and Dr. Chris Raczynski.

I would also like to thank my friends, my family, and my girlfriend, Kendell Worden, for a variety of aids too diverse and wonderful to lay out in this short section. I have come to learn that a doctoral research program is as much about mental toughness and discipline as it as about scholarly talent, and these would have been impossible to maintain without having such excellent people in my life.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF SYMBOLS AND ABBREVIATIONS

C2 ..... command and control
CAS ..... complex adaptive system
CDD ..... capabilities development document
DE ..... Directed Evolution
DFS ..... depth-first search
DoDAF ..... Department of Defense Architecture Framework
DSMC ..... Defense Systems Management College
EA ..... evolutionary acquisition
EO ..... Engineering Object
FBD ..... functional block diagram
FO ..... Functional Object
FSA ..... Functional Solutions Analysis
GA ..... genetic algorithm
GTE ..... Guided Technology Evolution
ICD ..... initial capabilities document
IDEF ..... ICAM Definition Languages
INCOSE ..... International Council on Systems Engineering
JCIDS ..... Joint Capabilities Integration Development System
JLC ..... Joint Logistics Commanders
MAST ..... Micro Autonomous Systems and Technology
MAV ..... micro air vehicle
MDA ..... Missile Defense Agency
NAV ..... nano air vehicle
P3I ..... pre-planned product improvement
SD ..... spiral development
SP2 ..... Strategic Planning and Prioritization
SV-8 ..... Systems View eight: Architecture evolution plan

SV-9 ..... Systems View nine: Systems technology forecast
SysML ..... Systems Modeling Language
TFDEA ..... Technology Forecasting by Data Envelopment Analysis
TIES ..... Technology Identification, Evaluation, and Selection
TIF ..... Technology Impact Forecasting
TOGAF ..... The Open Group architecture framework
TRIZ ..... Technique of Inventive Problem Solving (in Russian)
TV-2 ..... Technical View two: Standards evolution plan
UML ..... Unified Modeling Language

SUMMARY

In the development of systems with ever-increasing performance and / or decreasing drawbacks, there inevitably comes a point where more progress is available by shifting to a new set of principles of use. This shift marks a change in architecture, such as between the piston-driven propeller and the jet engine. The shift also often involves an abandonment of previous competencies that have been developed with great effort, and so a foreknowledge of these shifts can be advantageous.

A further motivation for this work is the consideration of the Micro Autonomous Systems and Technology (MAST) project, which aims to develop very small (<5 cm) robots for a variety of uses. This is primarily a technology research project, and there is no baseline morphology for a robot to be considered. This then motivates an interest in the ability to automatically compose physical architectures from a series of components and quantitatively analyze them for a basic, conceptual analysis. The ability to do this would enable researchers to turn attention to the most promising forms.

This work presents a method for using technology forecasts of components that enable future architectural shifts in order to forecast those shifts. The method consists of the use of multidimensional S-curves, genetic algorithms, and a graph-based formulation of architecture that is more flexible than other morphological techniques. Potential genetic operators are explored in depth to draft a final graph-based genetic algorithm. This algorithm is then implemented in a design code called Sindri, which leverages a commercial design tool named Pacelab.

The first chapters of this thesis provide context and a philosophical background to the studies and research that was conducted. In particular, the idea that technology progresses in a fundamentally gradual way is developed and supported with previous historical research. The import of this is that the future can to some degree be predicted

by the past, provided that the appropriate technological antecedents are accounted for in developing the projection.

The third chapter of the thesis compiles a series of observations and philosophical considerations into a series of research questions. Some research questions are then answered with further thought, observation, and reading, leading to conjectures on the problem. The remainder require some form of experimentation, and so are used to formulate hypotheses. Falsifiability conditions are then generated from those hypotheses, and used to get the development of experiments to be performed, in this case on a computer upon various conditions of use of a genetic algorithm.

The fourth chapter of the thesis walks through the formulation of a method to attack the problem of strategically choosing an architecture. This method is designed to find the optimum architecture under multiple conditions, which is required for the ability to play the "what if" games typically undertaken in strategic situations. The chapter walks through a graph-based representation of architecture, provides the rationale for choosing a given technology forecasting technique, and lays out the implementation of the optimization algorithm, named Sindri, within a commercial analysis code, Pacelab.

The fifth chapter of the thesis then tests the Sindri code. The first test applied is a series of standardized combinatorial spaces, which are meant to be analogous to test problems traditionally posed to optimizers (e.g., Rosenbrock's valley function). The results from this test assess the value of various operators used to transform the architecture graph in the course of conducting a genetic search. Finally, this method is employed on a test case involving the transition of a miniature helicopter from glow engine to battery propulsion, and finally to a design where the battery functions as both structure and power source.

The final two chapters develop conclusions based on the body of work conducted within this thesis and issue some prescriptions for future work. The future work

primarily concerns improving the continuous optimization processes undertaken within Sindri and in further refining the graph-based structure for physical architectures.

CHAPTER 1

INTRODUCTION

In modern Western society, technological development is appreciated as a vital path to fulfilling a wide variety of human needs. While there are a number of critics that allege (many with good basis) that technology may be overemphasized, there are few that would argue with the benefits of medicine, information technology, or mass production of articles to provide for basic human needs. In either case, a large number of technological projects are undertaken every day, whether by longstanding corporations such as General Electric, government agencies, or insurgent biotechnology startups.

With a focus on technological solutions to a wide variety of problems (and perhaps a tendency to invent problems to solve), there is inevitably a wide distribution of levels of ambition in technical projects. The ambitions range from simply improving the durability of a given dye or efficiency of a washing machine all the way up to revolutionizing our food system for greater nutrition, enabling people to move about the planet without endangering it through carbon emissions, or moving to inhabit the solar system. As the level of ambition goes up, so too does the complication of implementing solutions.

The US Federal government has become one of the most ambitious promoters of new technology via multiple agencies, including NASA, the National Institutes of Health, the Department of Energy, and Department of Defense. As the agency with by far the largest technology budget – $79 billion in 2008 [27]– the Department of Defense is worth examining for its priorities and methods in technology development.

The technology development process in Department of Defense has received a great deal of scrutiny over many years, with multiple reforms. The current mode of acquisition for the department is very mindful of technology development, planning for

all of its potential risks and delays, as well as the benefits that it boasts. There are a variety of very ambitious initiatives currently underway, including persistent radio contact [103], the ability to create a floating military base [72], and the ability to have persistent surveillance in an area, both in open fields [145] and inside caves and structures [111]. All of these goals require advances in the state-of-the-art, and often require many pieces of hardware to work together to achieve them. Because of both the required cooperation and increase in individual system performance, there must be rigorous and comprehensive methods for technological planning and management. Two methods in particular, evolutionary acquisition and spiral development, will be described later in this section.

One instance of a technology development project currently underway is the Micro Autonomous Science and Technology (MAST). The stated goal of this project is

> *"To enhance tactical situational awareness in urban and complex terrain by enabling the autonomous operation of a collaborative ensemble of multifunctional, mobile microsystems. [111]"*

It is currently understood that one requirement for reaching this goal is the ability to develop ever smaller robotic creatures in order to surreptitiously deliver sensors to a given interior space. This is in line with other miniaturization work, such as DARPA's Micro Air Vehicle initiative [143]. Many of the same drives – e.g., making components perform multiple functions and elimination of dead space – of that project are in line with those of MAST.

The MAST project is currently organized as a Collaborative Technology Alliance (CTA), meaning that multiple centers are responsible for developing an array of technologies that have the potential to advance the state of the art in micro-machines. Unlike the efforts to develop the next generation of aircraft or telecommunications device, there is no single system concept or morphology that guides the MAST research. Thus, there are a large variety of possible microsystems, each of which is the

combination of multiple technologies being developed by the consortium. The great variety of potential designs is illustrated in artwork shown by the project to popular media, such as the dragonfly pictured in Figure 1. Also, these technologies are being developed over a variety of timeframes. This further implies that there is a need to evaluate many different combinations of technologies with levels of performance that change over time.

Autonomous systems on the scale called for by MAST have many purposes envisioned for them. This means that there is the potential for a "Cambrian explosion," with a very wide variety of platforms, sensors, power systems, and strategies for use. All of this variety will likely lead to a handful of dominant forms, just as it has for many other technologies (e.g., aircraft, bicycles, motorcars) that have come before. Along the way, there are many choices to be made between competing forms, with the benefits of success accruing to those who choose the correct forms.

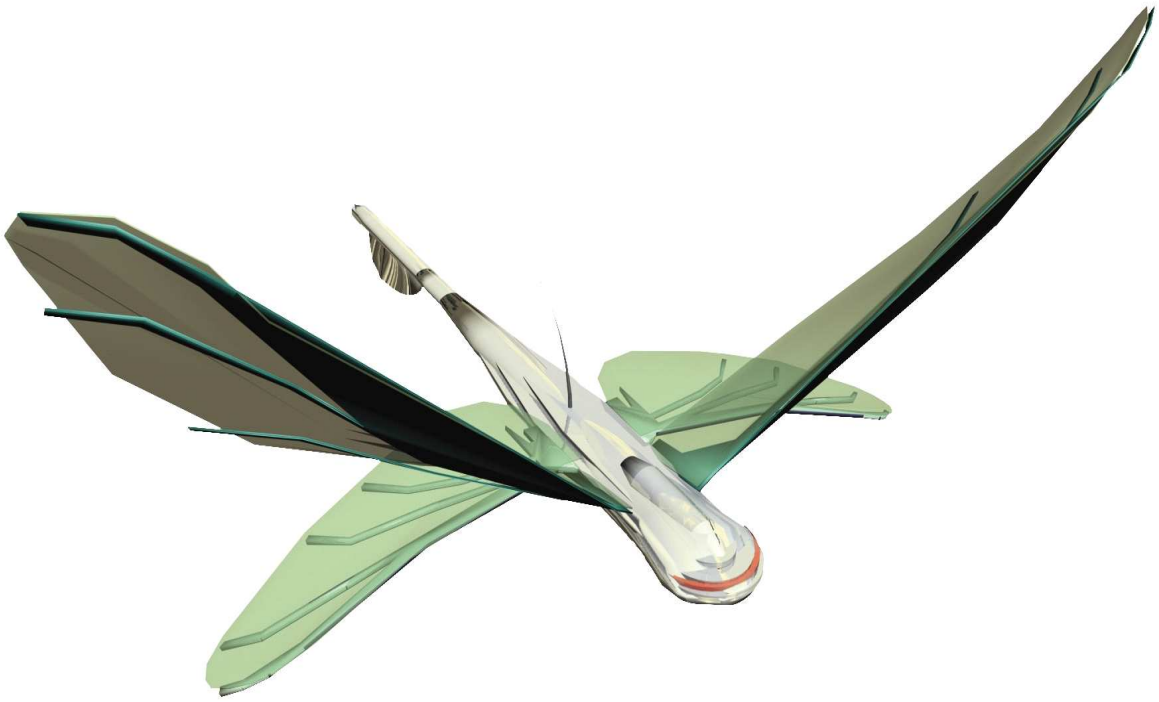


Figure 1. MAST dragonfly concept [79]

In addition, the MAST project also brings a special requirement to the system synthesis process. A design goal for the program is to promote the use and development of *multifunctionality*, the ability of a given subsystem to perform a task that was traditionally broken up and implemented in multiple, relatively independent subsystems. This means that simply assigning roles to given technologies and evaluating their impacts by the use of fixed metrics and multipliers will not necessarily work to evaluate all possible concepts. This will be a pivotal issue in formulating the tools developed within this thesis.

The motivating factors above lead this thesis forward into considering several aspects of the essence of technological progress, previous techniques for anticipating and managing that progress, and to provide a new take on the problem. By the time this thesis is finished, a new way of investigating the interplay between architecture and new technologies will be presented.

The choices between different forms, of course, are not constrained to this one program. The choices to be made between forms in the MAST project differ from other domains in number rather than type. The problem of selecting alternatives with the greatest potential is most acute with the large number of possibilities, but this problem is taken seriously throughout the Department of Defense. It is infused within its procurement procedures, in the requirements for Analyses of Alternatives and in the interest in a procedure known as evolutionary acquisition. The evolutionary acquisition highlights a great interest in forecast the effective life of a given architecture, and so will be used to further motivate this thesis.

Definitions of Technology

Before dealing with technological advance, it would be very helpful to understand just what technology is. The understanding of the essence of technology leads naturally to pondering how it comes into existence and how it advances.

W. Brian Arthur begins one of his papers [34] by examining the essence of technology. He defines technology "as a means to fulfill a human purpose." Further, he lists three principles to be mindful of while considering how technology changes:

1. A technology fulfills some expressed purpose – some need – personally or socially perceived.
2. A technology is built always around the reliable exploitation of some base phenomenon as envisaged through some principle of use.
3. A technology [often] requires other sub-principles (and therefore sub-components) for its practical working. In this it consists of components that are themselves technologies (and in turn consist of other technologies), the whole arranged in a recursive hierarchy.

The National Academy of Engineering provides another good explanation of the nature of technology [19]:

"Technology is the process by which humans modify nature to meet their needs and wants. Most people, however, think of technology in terms of its artifacts: computers and software, aircraft, pesticides, water-treatment plants, birth-control pills, and microwave ovens, to name a few. But technology is more than these tangible products.

"Technology includes all of the infrastructure necessary for the design, manufacture, operation, and repair of technological artifacts, from corporate headquarters and engineering schools to manufacturing plants and maintenance facilities. The knowledge and processes used to create and to operate technological artifacts -- engineering know-how, manufacturing expertise, and various technical skills -- are equally important part of technology."

Finally, the venerable Wikipedia has the following definitions related to technology [20]:

Science is the formal process of investigating natural phenomena. It produces information and knowledge about the world.

Engineering is the goal-oriented process of designing and building tools and systems to exploit natural phenomena for a practical human means. Engineers work within the constraints of natural laws and societal needs to create technology.

Technology is the consequence of these two processes and societal requests. Most commonly, the term technology is used as the name of all engineering products.

The key aspect of technology to be used in this research is that fact that it is a manipulation of natural principles to achieve an end. Thus, improvements in technology will necessarily involve improvements in the understanding of natural principles, whether by concerted research or by happy accident that is analyzed and understood after the fact.

Definitions of Architecture

Another term that will be used often in this work is architecture. As with technology, there are a wide variety of definitions stated, although they all coalesce on general principles of composition, interaction, and the basis for intuitive clustering based on "style."

Maier defines architecture [110] as "The structure (in terms of components, connections, and constraints) of a product, process, or element." He then goes further and determines a useful architecture as one that gives the system sponsor sufficient information to understand the system design's value, cost, and risk.

The INCOSE System Architecture Working Group defines a system architecture as "The fundamental and unifying system structure defined in terms of system elements, interfaces, processes, constraints, and behaviors."

A final definition is provided by the DoD Architecture Framework [23]. There, architecture is defined as "the structure of components, their relationships, and the principles and guidelines governing their design and evolution over time."

The relevant aspects of these definitions for this thesis are the emphases on component / element selection and the relationships between these components. These

relationships serve as signals to designers about what design challenges to expect as the concept moves through full development into finished hardware. For example, a jet engine architecture is typically defined by the elements of the engine (fan, compressor, combustor, turbine, nozzle) and the rules of interaction, namely air routing and connections between the turbomachinery. A mixed-flow turbofan will conjure a different image and set of design challenges than an unducted fan.

Evolutionary Acquisition and Spiral Development

Background

Evolutionary acquisition is a specific Department of Defense strategy for acquiring a system or an interacting set of systems. This strategy aims to transform very large technology goals – such as enabling daily access to space, building a lunar base, a swarm of artificial insects, or gaining the ability to swat down ballistic missiles – into manageable chunks. These smaller pieces are to be released over time, building one on another in order to reach the eventual goal. Often, many of the pieces can be built immediately and need replacement on different timescales. In addition to making the smaller pieces easier to manage, the strategy allows for some capability to be released into the real world as it becomes available, rather than being "nearly ready" for years at a time. This capability is released in small pieces via systems that are called "incremental releases." The time between increment releases can be used to advance the technology (which was insufficient to meet the large goal at program initiation) and to gather feedback from users in the field.

It is important to note that the evolutionary acquisition strategy is by no means a panacea for all systems. The strategy is meant specifically to be applied to systems that must greatly increase their levels of performance, deal with highly uncertain requirements, or both [87]. In the case that the goal is to greatly improve technologies,

the evolutionary acquisition strategy pursues incremental development, which means that the incremental releases are designed to integrate new technologies in sync with ongoing technology development projects. With requirements uncertainty, spiral development is used to reduce the risk of developing a system that does not address the end user's true needs.

The primary goal of the evolutionary acquisition / incremental development combination is to avoid tying up a hardware development program with simultaneous technology development. Relying upon unproven technologies was shown to be disastrous in several programs, including the Comanche helicopter [37] and the X-33 single-stage-to-orbit demonstrator [53].

It is also worth considering that "evolutionary acquisition" is the natural process of the private sector, especially with new technologies [71]. Rather than having a single customer, the government, private companies typically start small with a new technology, attempting to recover development capital through niche markets. Also, in cases such as cellular phone networks, private firms start in different locations and, through a series of agreements and cross-contracts, patch together a national system. Another private sector attribute is that the pace of competition is swift, meaning that new products must continuously be produced as new technologies approach maturity. While in some cases, such as the microprocessor industry, industrial rivals come together to try and understand the future, the commercial process of competitive technological evolution is more Darwinian than pre-planned.

Seeing that the commercial world was highly successful in bringing technologies into service in a rapid and cost-effective way, the Department of Defense began to experiment with evolutionary acquisition. It was first used in the command and control community, in order to develop networks of communication. This proved a natural fit for evolutionary acquisition, as networks are naturally made up of interchangeable blocks that can be upgraded over time to increase performance according to need. Over time,

the evolutionary approach has become more popular among the decision-makers of the defense community, and now it has become the preferred mode for systems acquisition [10].

Evolutionary acquisition focuses on the long-term, where multiple segments or releases of systems are managed to form a coherent goal [31]. Spiral development is more focused on the development of any individual increment release, and is meant to reduce the risk of developing the wrong system.

Spiral development is a method by which to schedule engineering work, and is most often compared to the waterfall model, where work proceeds linearly from conceptual design to final product. The spiral development plan builds proofs of concept and detailed designs of choice modules as a means to communicate with a project's stakeholders in order to negotiate realistic and useful goals. As goals are defined, the clear portions of the work are put aside (although they may be revisited if absolutely necessary) and the focus shifts to buying down remaining risk. This process is complete when all of the stakeholders have come to an understanding as to what is to be produced, and the group can proceed to finishing the product.

Spiral development emerged as an application for the software development community. At first blush, it may seem that it is best suited for its realm of origin since software is not a physical product. In other words, revisions can, in theory, be made relatively easily. However, in complex software, a well-considered blueprint is vital [22], just like any other system. Attempting to hack away at large software products to bring them to completion invites a large number of hidden bugs and growing pains down the road. Also, advanced simulation and rapid prototyping can bring down the cost of risk reduction measures for hardware, drawing the hardware and software worlds closer together. In both cases, forcing too much structure too early can eliminate opportunities that may reveal themselves as systems are implemented and tested. The key to both the hardware and software worlds is to understand high-risk requirements and their impacts

in the real world before committing to precisely defined values for these requirements prematurely.

The real importance of spiral development is in its appreciation of cost-benefit trade-offs. An example given by Boehm [46] is a government-funded information system. Early analysis led to the desire for a one-second response time for the system. Through the regular, linear process, it was discovered that this system would cost nearly $100 million, with this cost largely driven by response needs. After this episode of sticker shock, the developers built a special prototype, where the user interface and experience would be simulated. It was found that a four-second response time would satisfy 90 percent of the users, which resulted in a much more palatable $30 million price tag.

History

A short timeline of the development and application of evolutionary acquisition as defense policy can be found in the literature [146]:

1983 Armed Forces Communications and Electronics Association study Evolutionary Acquisition (EA) for C2 programs.

1986 Joint Logistics Commanders Endorse EA.

1987 Defense Systems Management College and JLC publish a guide for using EA on C2 programs [30].

1988 Boehm publishes original article on spiral development [45].

1990 JLC/DSMC recommends that EA language be included in DoD 5000 regulations.

1995 Defense Acquisition Executive issues guidance on the use of EA [87].

1996 National Center for Advanced Technologies recommends EA as preferred approach to acquisition.

1999 Section 912 (named for provision in 1998 defense spending bill) study team endorses EA approach.

2000 DoD publishes new 5000 regulations endorsing EA as preferred approach.

Methodology

Both evolutionary acquisition (EA) and spiral development (SD) are managerial processes, and so their primary mode of operation is through the definition of activity sequences, milestones, and conditions for proceeding from one activity to the next. The methodology discussed here will focus on how these activities are different from their more traditional (Cold War system) counterparts.

There are six pillars to evolutionary acquisition as described by the Joint Logistics Commanders [87]. The first is an *evolution plan*, which is to be an outline of how capabilities increase over time. The *architectural plan* is a description of the principles of the system architecture and what kinds of change it can facilitate. A *technology road map* is required to understand the available technology in the given timeframe. A summary of funding requirements forms the basis for the *funding profile and contract strategy*. User fielded verification is required in a *product assurance plan*. Finally, support analysis that determines the level of support a new system will require is to found the *integrated logistic support plan*.

For this work, four of these pillars will be considered, with the other two as out of scope. Product assurance planning and logistics are typically management activities. Logistics are potentially important, as they may serve as a constraint on possible designs, but would require a separate analysis from performance-based analyses. Integrating a logistics analysis with the methods eventually proposed in this work will be considered future work.

For evolutionary acquisition, the largest divergence from traditional acquisition is in the treatment of requirements via the evolution plan. Traditionally, a set of technical

requirements would be laid out, and system development would be undertaken. If technologies were to be matured along the way, so be it, as those that defined the requirements were usually aware (or made aware) of the consequences of their decisions. In the traditional view, half measures weren't worth having, possibly because the Cold War was as much about posturing as it was about actual capabilities. Also, specifications were traditionally developed in accordance to countering threats. In the case of developing a solution to match a Soviet MiG, an eighty percent satisfaction of requirements would likely result in a large number of pilots killed.

In the new view, partially meeting the ultimate requirements on a first try may be more valuable than spending a long time to deliver nothing. The key to this is that the partial requirements *must have military value* [85] to the end user in order to be considered. In cases where great leaps of technology are required, this is likely to happen. For example, being able to stop missiles in a local theater of operations (rather than over the globe) is a useful capability in its own right. Over time, the requirements are meant to become more and more ambitious, until the ultimate goals for the system are met. This type of evolutionary metric is called a time-phased requirement [137].

Time-phased requirements are supported in evolutionary acquisition by replacing the document for static specification, the Operational Requirements Document (ORD), with a pair of documents collectively focused toward continuous progress: the Initial Capabilities Document (ICD) and Capabilities Development Document (CDD). The ICD embodies both short- and long-term planning by laying out the needs of the user for a given mission [16]. The ultimate goals are laid out, in addition to the minimal requirements needed to provide a useful capability. The CDD is designed for planning out how individual system releases will conform to a more stringent set of requirements over time [16].

These requirements are meant to be derived through modeling and simulation [12], or even war-gaming, of new concepts of operations. All of this activity is meant to

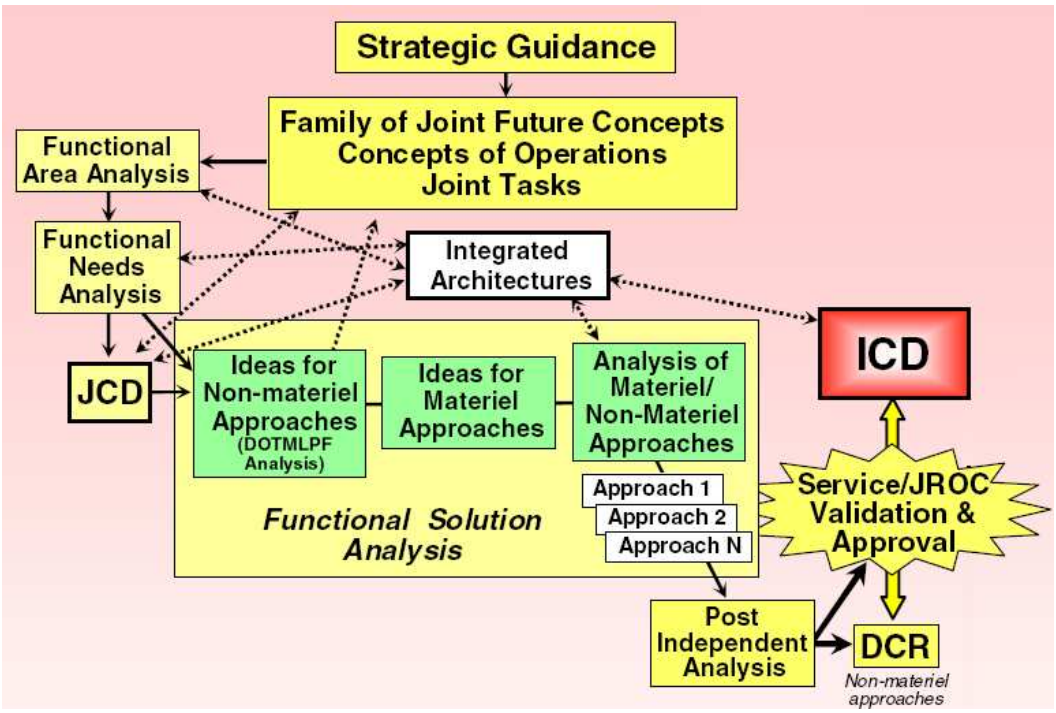


Figure 2: Flow of top-level analyses into requirements via JCIDS [12]

take place within the JCIDS process [7], which aims to base requirements on a functional need rather than overcoming a baseline threat. The flow of this process into ICDs can be seen in Figure 2. This modeling and simulation not only identifies gaps, but also is meant to be used to understand what the threshold operational requirements and desired operational requirements should be. These requirements will be used to book-end the time-phased requirements with a start and an end point, and later discussions will be needed to decide how rapidly to progress from one to the next.

The modeling and simulation methodology used to achieve what is described here is hardly trivial. This methodology is itself a field of active research [3], and is outside the scope of the discussion here.

Architectural plans are meant to capture either the intent of the systems designer to have an architecture that can grow with changing requirements or the ways in which the architecture will be changed to meet new requirements. This information is meant to be captured within the DoDAF framework under SV-8 and TV-2 [104], [123].

The SV-8 bears some further exposition. The System View 8 is intended to provide planners with a notional timeline of when the systems described in the architecture will be upgraded or replaced by new versions. The SV-8 should also illustrate what new capabilities a given upgrade or replacement will deliver to the architecture at large, and new requirements imposed on other systems, if any. The SV-8 is also meant to go hand in hand with the SV-9, which is a fully developed technology forecast for relevant parts of the architecture. The SV-9 is intended to describe both emerging capabilities from new technologies and trends in industry to migrate between relevant technologies. Thus, a fully formed DoDAF product will require a technology forecast. The methods of developing this forecast will be addressed in another section, later in this paper.

The timeline of progress is the province of the technologists and manufacturers, although there is certainly pressure from the acquirer to make this transition as rapid as possible. The CDD provides the amounts of requirements improvement that would justify the release of each new and improvement system. These improvements can be planned incrementally, when requirements are well-defined, to leverage the old pre-planned product improvement (P3I) process. Alternately, the spiral development process can be used.

The milestones of acquisition have also been modified for evolutionary acquisition. The start of a product line has two phases, Concept Refinement and Technology Development, which are not applied to later system releases [11]. The idea is that there is a separate effort required for dealing with the overall vision for system development, within which the various system increments are released. This further implies that, whereas stand-alone releases are self-contained and centered on the present, the evolutionary approach continually looks into the future, using testing and field results from previous stages to plan future steps.

Finally, evolutionary acquisition has a series of specific triggers for its use, as

described in the JLC manual [87]. The six areas identified are:

1. Uncertain requirements or requirements likely to change over time
2. Uncertainties in final system performance due to reliance on developing technology
3. Expected changes in the experience of a system's usefulness by its users
4. Need for an initial capability before planned final capability can be available
5. Budget uncertainties
6. Evolving needs in order to maintain compatibility with other systems

Spiral development is a much more hands-on methodology, since it is concerned with the actual delivery of a given product. Boehm [46] asserts that spiral development is a process model, meaning that it is intended to solve two questions: what the next activity should be and how long it should continue. Rather than using a prescribed set of phases and milestones for the whole project, the spiral development process depends on a series of risk assessments to plot the way forward. Further, Boehm identifies six invariants that can be used to determine whether a process is truly representative of spiral development.

The first invariant is concurrency of development. Several design artifacts are meant to be developed simultaneously: operational concept, system and software requirements, architecture, and key components. Some variances are allowed for the relative level to which each artifact is to be developed in each cycle, as well as the number of cycles needed. If architecture is very important (operation with other systems, for example), then initial cycles will likely focus on sketching out the architecture and buying down risk on top-level requirements. If the new system represents a technology advance, high-risk technology items will receive the main focus in early spirals. Overall, the goal is to continuously get feedback as to what design decisions mean for implementation, not necessarily to do everything at once.

The second invariant is that every cycle in a spiral must contain work which considers critical stakeholder objectives, analysis of alternatives, risk reduction, stakeholder review and a commitment to proceed with decisions. These steps may

receive differing weights depending on the progress of the project, and risk can be bought down through a variety of methods, including testing, simulation and prototyping. This is the part of the process that is intended to ensure that the product developed is in line with the priorities of the stakeholders.

The third invariant is that levels of effort should be developed based on risk assessment. Although determining the optimal level of risk reduction is hardly elementary, the spiral development process demands that it be attempted. In the example in Boehm's report referenced in this section, market risks of late release must be balanced against the market risks of releasing a product with major defects. The fourth invariant is similar to the third, in that it focuses on degree of detail of specification or design rather than work effort.

The fifth invariant relates to the commitment of resources to various activities. Three milestones are required in the spiral process, namely the Life Cycle Objectives, Life Cycle Architecture and the Initial Operational Capability. The three milestones represent a financial commitment to architecting and design, specifications of the system's full life cycle and operation of the system respectively. An analogy to stud poker is used by Boehm to show that these milestones are the opportunities to continue to buy into a system's development or to withdraw support and terminate the project.

The final invariant is that a spiral process should look to the system and life cycle at-large as much as possible. This is hardly unique to spiral development as many developers have attempted to step beyond thinking of their job as finished once the product leaves the assembly line.

The methodology of spiral development addresses the conditions under which a product can progress through the development cycle. Since it is risk-focused, the active fields of research for spiral development include the estimation of risk and uncertainly, and in understand how much this is reduced through actions such as simulation and testing.

Problems Presented by Changing Architectures

All engineering is a mixture of theory, pragmatic adjustments, and insights gained from experience with a given machine. Most engineering practices are based on the principles of reductionism and the development and test of components in isolation from one another. Systems are recomposed steadily from basic parts into assemblies and components and then subsystems, with a final systems integration and battery of tests.

Since technological systems are designed and developed as compositions of individual pieces rather than as a whole, surprises during development are inevitable. Over time, experience renders these surprises into engineering best practices. A great number of these practices relate to the proper integration of components and identification of important cross-effects, such as the vibration given off by a moving part affecting other moving parts within the system or loosening fasteners and connections. These cross-effects may be predictable, but as the number of components in a system grows, the number of potential cross-effects increases exponential. Empirical trial-and-error is often a more sure and suitable method of discovering these effects than endless analysis.

According to the definition of architecture given above, a change in system architecture also changes the interactions between its elements. Many practices and rules of thumb are rendered void due to this, and many new mistakes will have to be made in order to master the new architecture. There are many examples in competitive industries where a previously unbeatable player was unable to master a new technological architecture. In cases where the old architecture become far less profitable than the ascendant replacement, these players suffered greatly in the transition.

However, it is not guaranteed that a shift in dominant architectures will render the competence of the previous dominant practitioner obsolete. Tushman and Anderson, in their paper on technological discontinuity cycles [33], show many examples of shifts that actually draw upon previous experience. Understanding what the old and new dominant

players will look like will certainly help a practitioner understand which case the transition will embody.

Thus, although a new architecture may open new levels of performance or enable abilities not available in previous architectures, the cost of transition is the abandonment of a body of familiar practice. This can be a rather steep cost, and so transitions must be taken very seriously and with great care.

Motivation for New Work

A major source of motivation for this work is the large number of possible architectures presented by radically new technology programs such as MAST. Beyond the sheer number of possibilities, the effectiveness of current methods is limited by special characteristics of the problem such as multifunctionality.

The problems of selecting between architectures does not merely exist when one is trying to figure out where to start, as in MAST. Selecting between architectures can be just as critical a task when deciding to leave a body of current practice for a new technological basis. This choice often presents great risks on either side. Those who favor the incumbent for too long can be trapped and fall from dominance. Those who move without sufficient preparation may never find their way in the new realm.

When planning for technological development, it is important to consider the limitations of currently existing architectures. The transition between architectures due to the changing properties of components has been investigated before by Hollingsworth [90]. However, this work focused on changes in requirements rather than technology's evolution over time. At any point in time, there are a great number of promising alternatives. The promise of these alternatives often rests on the development of a small number of critical technologies.

A good example of this is the case of remote controlled aircraft. The potential of miniaturization of electrical motors is greater than that of their internal combustion

counterparts. However, the energy density of batteries is far inferior to that of hydrocarbon fuels, greatly limiting the endurance of electrical aircraft. It is only recently that battery technology has progressed to a point that an electrical architecture has been capable of useful flight times. But once this has happened, electrical aircraft (especially for indoor and very casual use) have exploded in popularity. The Micro MX-1 toy helicopter [80] fits comfortably in the palm of one's hand, while typical internal combustion counterparts have reached their limits of miniaturization at larger sizes, roughly one foot from end to end.

It can be seen by the considerable work put into the development of evolutionary and spiral methods that planning for technological development is considered a major priority by the Department of Defense. Further, the body of work behind evolutionary acquisition stresses the importance and the co-dependence of both technology and architecture.

The reason for presenting MAST, evolutionary acquisition, and more general consideration of the risks involved in making investments in a new architecture is to show that the problem of architecture selection touches multiple realms. In each of these realms, there are high stakes, and technology development is considered to be of great importance to making the proper choice. Thus, the problem of selecting an architecture, predicated on current and future potential, is both important and sufficiently general to motivate further investigation.

This dissertation serves as a response to the problem of specifying an architecture while being mindful of its potential, which may hinge on technologies that have yet to be invented. The response to this problem, which is developed over multiple chapters, is a method for discovering the best architecture at a given time and level of technological progress. It combines techniques from combinatorial optimization, technological forecasting, and visual analysis to guide the would-be architect in making the critical decisions that have been alluded to in this chapter.

CHAPTER 2

LITERATURE REVIEW – PHILOSOPHICAL BACKGROUND AND STATE OF THE ART

Technology has been studied and discussed in a wide variety of fields. Its progress and development is a fascinating story on many levels: as a story of human drama and competition between firms, as a question of the creation of novelty, as the engine that drives our economy, and as an inspiration to future designers, among several others. Technological studies range from broad pronouncements of general trends and principles to empirical investigations of the progress of individual technical metrics over time.

In this chapter, some of these views are surveyed. The more general views are presented in order to give a scholarly treatment of the nature of technology, and then practical techniques for forecasting and roadmapping are presented. These are the collective readings that have guided the general development of this thesis and the research program that it describes. This chapter is presented to the reader in order to provide background that will be drawn upon later in the dissertation in order to build up the method that will eventually be presented.

The first group of ideas are those that relate technology development to the biological processes of innovation and experimentation. The insights from these readings ground a particular view of technological development, namely that it is a gradual process driven by improvisation and extension of existing bases of knowledge. In this view, there is no radical innovation in the same way there is no special creation in biology, although there can be extraordinary accidental discoveries. This view is laid out most strongly by Basalla. Although this collection of readings may appear at first to the

reader to be superfluous, they combine to form a bedrock of philosophy that has guided the development of this work. For if it was true that radical innovation existed, it would mean that new architectures could arrive totally without antecedent. Further, this would mean that it would not be possible to develop certain architectures from a collection of pre-existing (although possibly cutting-edge) components. At this point, any combinatorial approach to the problem would be fundamentally incomplete.

The next set of ideas belongs to a body of practice known as TRIZ. They also endorse the technology as a gradual process viewpoint, although in a more rigid framework than the first group of ideas. The body of TRIZ work is essentially a database of patents, from which a canonical set of inventive "moves" have been extracted to both improve the creativity of inventors and also to generate a type of forecasting method. In addition, the body of TRIZ work contains some interesting thoughts on the nature of component integration via its Substance-Field analysis.

The final set of ideas presented in this chapter are the state of the art in technology forecasting. The reason for looking through this body of literature is that at some level, the system to be designed has reached the maximal amount of decomposition, and the increasing performance of those atomic parts must be forecast into the future. "Atomic" in this context will have many different meanings depending on the number of architectures to be examined at a given time. In the scope of this work, the architectural search will be assumed to be conducted at the conceptual stage, and so relatively abstract representation of subsystems and components will be appropriate. As with usual conceptual practice, the level of performance of these components will be taken as estimates benchmarked upon past and present values. With the aid of technological forecasting techniques, future values can also be estimated. The body of technology forecasting knowledge thus serves as a base for existing tools to recruit into the methodology to be developed within this thesis. The most relevant work is that on quantitative modeling and extrapolation from previous technology data.

With the rationale behind the presentation of each body of background knowledge established, it is time to move through the material.

Darwinism as a Design Algorithm

Darwin's theory of evolution has been a tremendous success in explaining the biological world. Its power is so great that it is understood to be the central principle of modern biological theory [63]. Its success is in explaining the incredible variety of life and for giving mechanisms for the development of highly adapted and complicated structures found in nature.

It is not surprising that thinkers began to attempt to use this concept in other realms. The strongest position along this front is that currently represented by Richard Dawkins, Daniel Dennett and Susan Blackmore. In their collective view, a blind evolutionary algorithm develops human culture. It should be noted that as a philosopher and a psychologist, respectively, Dennett and Blackmore have spoken much more on the cultural Darwinism than Dawkins, who tends more toward biology.

In Dennett's words, the ideas of Darwin are a "universal acid" [60] that can be used to cut away all requirements for a designer in design processes. He discusses the idea of a universal algorithm that distributes the design process into a massively parallel one by use of agents. The only requirements of this algorithm are that the agents can replicate with modification, inherit traits from previous generations, and that they are preferentially selected for their traits. In this scheme, biological evolution is simply a specific case of a more general template.

Blackmore's *Meme Machine* [43] focuses on the evolution of culture, and goes so far as to suggest that memes are as powerful a unit of selection in society as genes are. Her argument proceeds from the notion that as people, we are often far more engaged in imitating (copying the memes of) each other than coming to fully rational decisions as to how we should proceed in a variety of situations.

Blackmore's emphasis on memes implies that any cultural practice, presumably to include the production of technology, is the ultimate result of meme processing. The brain takes in memes (inheritance from the outside), processes them (usually modifying slightly rather than radically) and then sends them back out into the world for reception (selection by how popular they become). This process is shown to compete with genetic drives in multiple examples: a memetic explanation for altruism, where the altruistic are better liked and thus better copied; the spreading of urban attitudes about family by cool (and unencumbered) single women, which lowers birth rates; and the growth of human brains far beyond their survival value since brain tissue is very energy expensive. The existence of this competition and its ability to drive suboptimal solutions (if only genes are taken into account) is her strongest evidence for the existence of memes.

Meanwhile, Dennett and Dawkins [59] both discuss the Darwinistic algorithm and its ability to navigate design spaces and fitness landscapes in general. Dennett uses the concepts of the Libraries of Mendel and Babel, which contain all possible genomes and English language books of a given length respectively. Included in the mythical Library of Babel is every living person's biography, in addition to false biographies that differ by a single detail, as well as by every possible typographical error. The Library of Mendel is used to illustrate just how vast a design space biological evolution has navigated with the sacrifice of untold legions of unfit creatures. Dennett takes his message further when he asserts that the Darwinian algorithm makes the design problem massively parallel, enabling it to accomplish incredible feats of design that truly would require a powerful intelligence to construct otherwise.

Dawkins' discussions of fitness landscapes focus mostly on the biological. However, he delves into the development of specific features such as the eye, which has evolved independently in multiple creatures in a wide variety of circumstances. In the story of the eye especially, which is illustrated in Figure 3, the progressions have a distinctly technological feel to them. The eye starts as a primitive light-sensing device,

with a simple distinction between light and dark. It then begins to acquire a concave shape so that it can detect the directionality of light to a limited degree. As evolution continues, additional structures begin to come into play, further refining the directionality of light detection and beginning to enable primitive images to form. Modern eyes have specialized sub-structures for enhancing images and enabling the eye to focus at multiple distances, as well as different types of photo-receptors (rods and cones) to develop a composite image with sensitivity to differences in light wavelength.

The eye is sufficiently complicated and well integrated to be mistaken as an intentional design by creationists, and offered as an example of a system too complex for evolution to generate. It is a worthwhile example for considering how complex systems

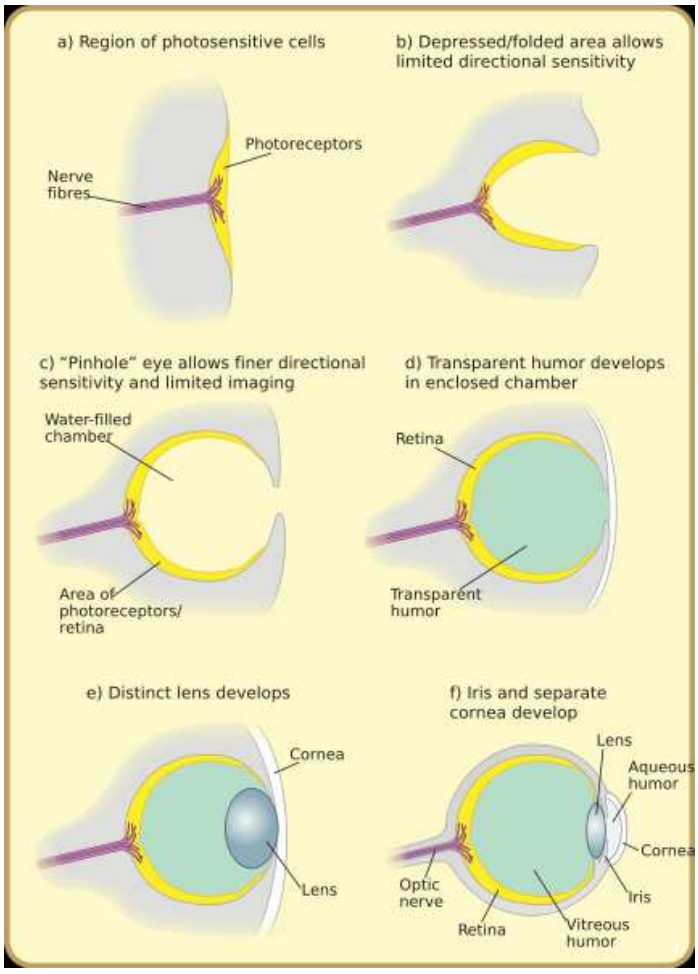


Figure 3: Evolution of a notional eye [117]

can be developed by bits and pieces. This example is instructive in another way, and that is how a functionally-focused evaluation of a structure may obscure the ability to see developmental paths. If the various parts of the eye are seen only in their current functionality in forming images, it is difficult to see how substandard eyes are useful. If, instead, the physical structures (photoreceptor, curved depression, protective fluid) are considered, then alternative and useful functions can be conceived. Through this conception, an evolutionary path can be identified. The story of natural evolution is a story as much about repurposing as it is about refinement.

Further, in *The Extended Phenotype* [58], Dawkins provides a trio of basic requirements for an evolutionary procedure to operate: replication, interaction and lineage. He attempts to develop a chain of logic that shows that human creations, such as cultural norms and technology, are also the products of evolutionary processes that operate within and between people. Even if the meme theory is not strictly true as a process for human thought, it can provide useful insights into the creative process, especially for very structured creations such as technological artifacts.

Complex Systems and Innovation

The Santa Fe Institute has made innovation in complex systems a standing research topic. This topic is further broken down to include innovation in biological and in technological systems, showing that the Institute views them as similar processes, albeit with domain-specific features. This work continues the world view and cross-fertilization that characterized their workshops on the economy as a complex system [1], [2], which mostly concentrated on finance and macroeconomics without much discussion of technology itself.

A major contributor to the field of complex systems is Holland, who invented the genetic algorithm and worked on building the field of complex adaptive systems (CAS). Before the concept of CAS was widely introduced [88], Holland co-authored book on

induction [89], including its usefulness in artificial systems and how it related to scientific discovery. Psychologists and a philosopher of science supplemented his expertise in artificial systems.

In the book on induction, the basis of thought is considered a goal-seeking rule, usually of an "if-then" form. These rules are constantly evaluated by experience, either increased or decreased in strength. Exceptions are carved out and rules are merged and bundled to form useful frameworks and heuristics for default approaches to new problems. These rules are also built into hierarchies, moving from rules of broad applicability to those governing specific situations.

Various inductive processes were considered within the rules framework, including generalization and analogy. These mechanisms are of special interest to the scientific process, and are put together on that chapter. Analogy in particular was in interest, as both the rules for transferring knowledge for one domain to another, and the decision of domains to search for analogous rules are important questions. Every application of one rule set to a problem is described as a transformation from mental model to the real world, or vice versa during learning.

The thrust of the work in the scientific section is the authors' stated belief that the proper formulation of a knowledge model would need to allow for interchange of parts, rating ideas, and rational maps between old and new problems. For the authors, multi-part rules ordered into hierarchies answered these requirements. At the time of their writing, computer systems to implement these rules were still new, but showing promise in developing novel rules on their own. The rules-based technique is now the foundation of the expert systems approach to artificial intelligence.

One fascinating study on technological invention by artificial systems was undertaken by Arthur and Polak [36], where the authors simulated the evolution of circuits. Most of these studies apply genetic algorithms to evolve new basic circuits, but this particular example added a twist. As "basic" circuits where identified that solved

niche needs (n-bit adders, multiple-bit gates), they were packaged by the algorithm and treated as new fundamental parts which could be added to the circuits as a whole. This added an element of bootstrapping to the algorithm, brought much greater speed to the algorithm and more closely approximated modern design processes.

Arthur also spends some time writing on invention, asserting that something beyond simple Darwinian variation [34], [35] in fabrication procedures is needed to explain major technological leaps. As he says, "vary 1930s radio circuits all you want and you will not get radar." He uses his definition of technology as the exploitation of natural phenomena to human purpose in order to tease out its logic. He sees invention as the marriage of a new physical effect to a purpose, whether the purpose itself is new or not. This either arises due to the impetus of a specific need, as was the jet engine, or as a serendipitous extrapolation from observing a phenomenon, such as X-ray imaging.

Another look at emergence of innovation from complex interactions comes from a direct biological example. Fontana [75] writes of how innovation has a different meaning at different levels of biological expression, from the gene to the full regulatory network in a living creature. Innovation on the gene string can be trivial (simple shuffling of the sequence), but can be profound once the many interactions of the network are taken into account. A network of chemical processes in development is much like a fabrication process in this view, and the appropriate network can develop chemicals never before seen. In this view, a "trivial" change can emerge as a profound new function within a life form. Further, there are a series of different RNA sequences that develop shapes that are functionally equivalent, leading to a "neutral network" of gene variants that can span design spaces and make populations have more latent variety than would appear to selection. He describes the networks as "high-dimensional sponges that are heavily entangled with one another" that provide access between one RNA shape and another without being subject to selection pressures until a sequence change causes a jump between networks.

Another important body of work in this area is that of Kauffman, particularly on correlation of fitness landscapes. His NK model [95] is widely used in computational studies, and has been applied to biological and technological fitness searches. N represents the dimensionality of the problem, and K represents the number of other nodes that each node n depends on for its fitness level. The K parameter is used to vary the amount of interaction between different genes in creating fitness. When $K = N - 1$, the landscape becomes a field of noise, and when $K = 0$, the landscape is a simple one with only one maximum.

The fitness landscapes gave way to results that determine statistically optimal distances for firms to search in the space (how many variables to change at once) in order to find improvements based on the correlation of the landscape, cost of search and current position on the landscape [96]. In the firm search formulation, each technology is represented by its generative recipe (a sequence of fabrication steps) with each step tunable to various settings. Steps can interact, making each other either less or more efficient. The goal of the problem is to find the most productive recipe. The results given in the study referenced in this paragraph showed a rise in optimal distance as search cost and initial fitness decreased, as one might think. The steepness of that rise increased with increasing correlation of the landscape.

If fitness landscapes are easier to navigate as interactions (K) decrease, this implies that a major architecting goal should be to reduce interactions within a system. But this is exactly what systems engineering does, by separating subsystems and minimizing their interactions. Thus, its procedures form a process that accelerates innovation.

The complex systems view of the evolution process is that it is the by-product of goal-seeking systems. This approach heavily uses computer simulation, which allows it to demonstrate very interesting dynamics within networks and agents that attempt to improve their current situation.

Evolutionary Economics Views on Technology

Background

The field of evolutionary economics is currently considered heterodoxy in the economics community, with the neoclassical school as the orthodoxy. Evolutionary economics is currently attempting to show that it can deal with issues that neoclassical theory has so far failed to explain, such as the role of technology, failure to converge to equilibrium, and so forth. A defining feature of evolutionary economics is the "evolutionary" portion, where the main point of interest is how systems and structures arise organically from the complex interactions of a variety of agents.

The neoclassical view of the world is based on mathematics and physics, with a strong emphasis on the results of equilibrium. In the neoclassical view, the attractors of matching supply and demand are sufficiently strong to make it possible to ignore most forms of non-equilibrium behavior. Further, there is a specialized class of actors, known as arbitragers, which serve to sweep up any mismatches between real value and asking price to their own profit. So, even if a market is not perfectly rational or brilliantly executed by all, the highly trained would serve to converge it to its ultimate equilibrium.

Where evolutionary economics steps into the larger discussion is to attempt to explain how the structures of neoclassical economics arise from a group of interacting individuals, and how the results of these interactions may diverge from the idealized results. Further, the neoclassical school treats technology as an external variable, while the evolutionary school attempts to internalize technological progress. Since technology is the engine through which material wealth grows as greatly as it has, it is clear that a formulation of economics that cannot deal with it contains a very large hole.

Although evolutionary economics has not solved the technology problem, it is deeply interested in developing a model of technological progress. This model is also meant to understand what causes technology to progress at a given rate. Thus, several

economists have now jumped into the field of understanding technology along with the biologists and complex scientists.

The first appreciation of the role of technology in economics is often attributed to Schumpeter [139]. He wrote of the entrepreneur and his role in breaking out of economic stasis to drive growth. He mostly emphasized the role of innovation in fighting falling profits and increasing productivity. Over his career, he made different arguments as to whether large firms or the entrepreneur was the best agent of creative change.

Technological evolution did not receive major theoretical work in economics until Nelson and Winter produced their model [124] decades later. The NelWin model, although generically focusing on R & D investment as allowing for a probabilistic "draw" of productivity and profit improvements, is regarded as a foundational work in evolutionary economics.

Despite a great deal of research on trading strategies and so forth, it is only very recently that any concrete proposals on modeling technology have been made by economists.

Technique and Scientific Spaces

Mokyr defines the replicator for technology as the technique, which is "a set of instructions on how to do something that involves production" [122]. There are two spaces that he keeps in mind, λ and Ω, which contain the spaces of feasible techniques and technical knowledge, respectively. Ω can contain everything from detailed and settled scientific theory to a very primitive knowledge that a given technique is effective. Mokyr points out that many techniques have been developed in human history based on nothing but trial and error.

Thus, there can be thought to exist three levels of selection in this scheme. The most basic is the usefulness of a given product or service with their quality, performance and cost properties. Above that is the usefulness of a given technique for production,

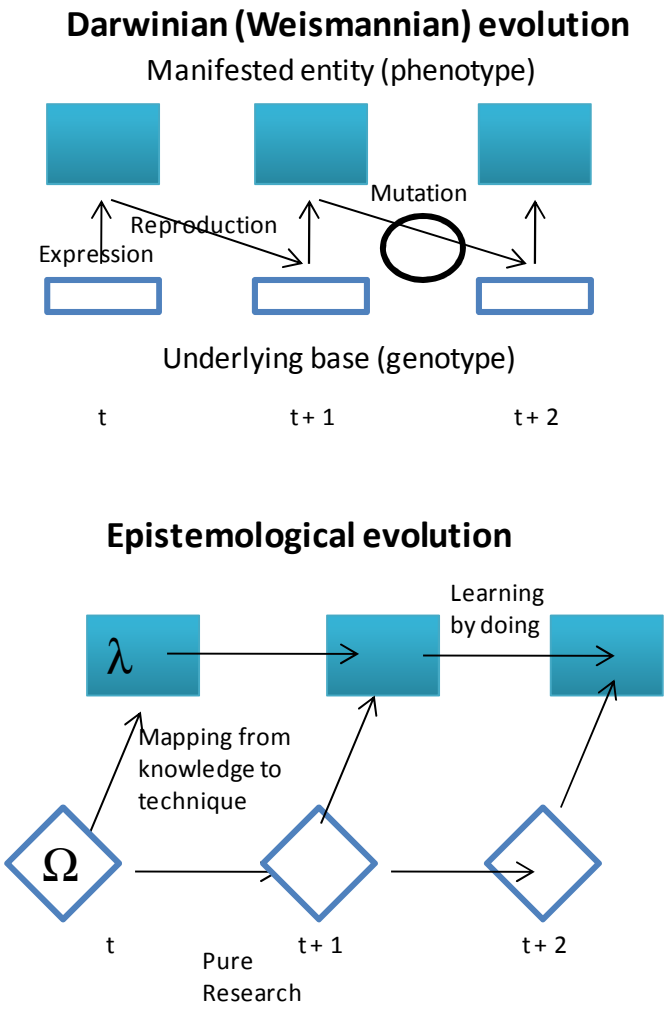


Figure 4: Recreation of Mokyr's diagram showing difference between technology and biology

which imbues differing levels of quality and cost upon the product, and may also imply such things as complexity of an organization required to support execution of the technique. The field of knowledge itself contains a competition for survival by various scientific theories and explanations. Another interesting aspect of selection is Mokyr's idea of "tightness" [121], which describes how closely a piece of knowledge and a technique are tied. The use of a given technique may yield appreciation of the physical effect it leverages, or an effect may need to be well known to translate into a quality technique.

The two domains, λ and Ω, both mutually interact with one another, where technique exploration can expand knowledge, and new theories suggest new techniques. This is different from biology, where adaptation input flows in one direction (organisms cannot change their genes, they can only carry them). Mokyr draws a figure of this process, which is recreated in Figure 4 in order to show the difference between genetic and epistemological evolution [120].

Finally, Mokyr examines the amplification part of the Darwinian framework. He discusses how a technique is "born" every time it is performed. This performance could be execution or teaching to another practitioner. As the technique is used more often, and is performed by more people, its frequency relative to the total population of techniques being performed increases. Thus, as a technique receives favorable results from selection, it is likely to be shared and used more frequently as a solution to the problems its users run into.

Physical Technology Schema

Beinhocker devotes a subchapter of his book, *Origin of Wealth* [40], to the place of technology in evolutionary economics. He describes the Library of All Possible Physical Technologies, which contains the full combinatorial space of potential technology-producing processes. Each of these processes is described as being a series of "if-then" type statements that combine to physically build a technology. This viewpoint is akin to the idea of the Von Neumann computer, where a simple set of logical comparisons, namely the AND, OR and NOT operators, combine to form the great diversity of modern computers.

The description of potential technology space is an indirect description of a process version of a Morphological Matrix. An example on the building of an axe describes the different combinations of materials, handles, sharpening rocks and the like that are developed and perfected over time. The various combinations of the

Morphological Matrix in these ancient times were explored by trial and error.

Beinhocker also [40] describes a "schema reader," which is a mechanism to transform coded information into the physical entity it is supposed to describe. In terms of technology, this would be an individual (or, in modern times, a series of individuals and teams passing the product from raw material to finished specimen) that understood what a given schema meant. For example, even if a full blueprint of 22nd century super-weapon landed on the Defense Secretary's desk, it is unlikely that early 21st century engineers could produce the weapon. It would be necessary to educate the engineers on 22nd century manufacturing and materials processes. Thus, Beinhocker describes an interesting co-evolution between technology and the skills of the people that implement it.

The schema reader view is evocative of skill webs [125], which provide an interesting problem of their own. The technological prowess of a society can be thought of as bootstrapping itself, much as the first primitive life did. As more complex forms evolve, new entities are needed to maintain and operate them. Thus, new technologies can be thought of as building new support structures (and new careers) as they go.

One aspect of schema reading that Beinhocker does not deeply explore is that modern technologies are hierarchical. In other words, the schema for a highly complex system, such as a space vehicle, requires the combined efforts of a variety of professionals to fully decode into physical hardware. Each professional can read a part of the schema that is in their core competence, work to understand a part that is adjacent to their core, and likely be lost on more distant portions. Also, when constructing a new schema, multiple experts will be necessary to construct it together.

Technological History

Another pivot point on the discussion of technological advance as a continuous vs. a discontinuous process takes place in historical studies.

An excellent walk-through of the history of technological development was written by Basalla [39]. In his *Evolution of Technology* book, he spends some time directly confronting what he regards as the myth of heroic / genius innovation. His first target is the account of the turbojet engine. Constant's study of the propeller to turbojet transition makes the case that there is a major technological leap between the two technologies, since there is very little resemblance between them [55], while alluding to the historical sources the jet engine draws from. Basalla interprets the connection between the turbojet engine and a rich history in turbine, internal combustion, and propeller design as evidence of the fact that even the most celebrated inventions are tightly connected to historic predecessors.

Two of history's most important inventions, with truly revolutionary effects, are put to the test as discontinuous leaps of technology. The Newcomen steam engine, the basis of Watt's version, can be seen to be the combination of elements from multiple epochs and civilizations, connected by an extensive investigation by Needham. Basalla deals with the transistor himself, linking it in a chain of electrical devices starting with the crystal rectifier used in radio sets. These sets were replaced by vacuum tube sets (which were themselves a spin-off of Edison light bulb fabrication techniques). However, germanium crystal rectifiers gained new interest when they were shown to be able to detect microwaves when vacuum tubes failed to do so. A great deal of study on the operating principles of the rectifier led to the ability to utilize it as an amplifier. Finally, a body of vacuum tube knowledge was used in the process, which is the source of the terms base, source, and emitter.

Basalla also highlights the existence of transitional vestiges, which exist in artifice just the same as they do in nature as signs in evolution. These vestiges are known as skeuomorphs in archeology, and are a common phenomenon. The first iron bridge had dovetailed joints and connected by keys rather than rivets, showing vestiges from woodworking methods. Modern plastic artifacts were designed to mimic their metallic

counterparts at first. Stepping outside of Basalla's work, it is also interesting to consider the "black metal" design practices that were associated with early composite use in aircraft.

Another look at technological heritage by Levinthal [108] investigates the development of several modes of wireless communication, including the quirks of history and supporters that helped them flourish. He breaks this development history into four epochs, with greatly differing requirements. The first generation of technology was driven by Hertz, who required control of frequency and electrical properties of his antenna for scientific reasons. Wireless telegraphy added a requirement to send these tuned waves over ever greater distances to ships at sea. Voice broadcasting then imposed a greater need for clarity and sensitivity of the receivers, and capitalized on improvements in that area on ship transmission. Finally, mobile telephony was driven forward by police agencies, which required portable receiver (and then transmitter) sets to intercept criminals in automobiles. It is also noted in the paper that mobile telephony technology was infused with wired telephony technology (AT&T repeaters) and broadcast technology (frequency modulation). Viewed independently, telegraph, mobile communications, and broadcasting may appear to be very different inventions, but they follow a very clear and historic pattern of lineage.

Summary of Evolutionary Material

There is a sizable body of scholarship that indicates that even the most revolutionary advances in technology are due to variations upon pre-existing inspirations, whether in other technology or in nature. If this is true, then biological evolution ideas, which are also based on improvisation and gradual improvement, can be applied to technological evolution. These ideas have been applied to technology from multiple directions, with sociological, complexity, and economic views presented in this chapter. Each of these perspectives reinforces the gradualist paradigm and provides insights about

its nature.

The material presented above brings considerations of human and natural innovation into close proximity. This is intentional, and it is meant to show commonalities in philosophy, which are corroborated by historical studies of technological innovation. For example, many stories of natural evolution, such as the eye, heavily involve the repurposing of existing structures. This evokes Arthur's definition of technology as a pairing of a physical phenomenon to a human purpose. It also recalls the stories of repurposing components that are told by Levinthal and Basalla. Pairing these ideas with the fact that computer programs and circuits have been successfully evolved from modular components through genetic techniques provides a very strong indication of the fruitfulness of an approach that applies evolutionary processes to pre-existing components. This will be carried forward in the next chapter as part of the philosophical basis for the development of the method and the various research questions that will be related to it.

The material on evolutionary economics was included to provide a potential direction for future leveraging of the work undertaken in this dissertation. Evolutionary economics speaks of the co-evolution of technologies and technology practitioners, and between science and technology. This is interesting from a purely academic standpoint, but also in considering the problems of future architectures. The reason for this is that in the cases of cutting-edge technology, the pre-existing components to repurpose may not exist in the technological world, but rather be taken from scientific apparatus. For example, the radio equipment used by Hertz for experiments was soon repurposed as components for transmitters.

The diverse body of reading presented in the preceding section should also be an indication of the fact that many of these ideas have yet to coalesce into a single field of investigation. Rather those of various viewpoints poke at the problems of technological

progress from multiple angles. It may thus behoove those interested in leveraging or extending this work to look into this reading.

TRIZ

Background

TRIZ (Russian acronym for "Theory of Inventive Problem Solving") is a body of methods, tools, and techniques that were designed to help inventors with solving creative problems. The foundation of TRIZ can be thought of as a set of powerful guided brainstorming methods. The original methods of TRIZ will not turn anyone into an inventor, but they can be very useful aids for designers in the process of generating innovative ideas. As TRIZ was further developed, its practitioner community became more and more ambitious in the problems they have attempted to tackle.

TRIZ has a solid foundation in Russian academia, but the main method of knowledge transfer to industry has been through consultancies. The various companies – Ideation, CREAX, InventionMachine and others – boast client lists that include many major engineering firms [26], [25], [24].

One exciting possibility that has been investigated for roughly the past decade, is that, if the rules of invention are general, they lead naturally into a theory of evolution for technology. In the same way that the various gene shuffling mechanisms can be used to understand potential directions for biological evolution, the directions of technological innovation can be used to lay out potential paths. For example, TRIZ contains a recommendation to hybridize technologies to mitigate weaknesses and compound strengths. This might lead to an evolutionary law of the following form: when two systems in different niches approach their maximum potential, these systems might be hybridized into a new system. The work leading to this point of view is presented here.

History

The originator of TRIZ was a Soviet inventor named Genrich Altshuller [4]. He is reported to have received his first patent at the age of 14 and was respected by his peers as a gifted inventor. Since he was often asked for advice, Altshuller decided to begin searching for systematic techniques to help people in solving their creative problems. This search started in the late 1940s. Much of what he found were psychology-based tools to "jump-start" the mind such as brainstorming. But, what the inventor desired was a way to understand how to measure the quality and rate of invention [118]. The search for understanding would lead to a variety of insights that would eventually find their way into various methods of TRIZ.

Altshuller was employed in the Soviet patent system, and so he had access to a very large amount of inventive data – the patent application base. He applied a scoring system to each patent based on how great an advance it represented within its field. A score of one was a routine invention created by someone trained in the field, while a score of five was a truly revolutionary design. In addition to understanding the magnitude of invention, Altshuller was also interested in developing an abstracted series of analogues that could be used to transform a seemingly intractable problem into one that could be dealt with. Altshuller worked mostly alone until 1980, when the first TRIZ specialist conference was held in Petrozavodsk, Russia. The body of work that was principally developed by Altshuller is typically referred to as Classical TRIZ.

As the TRIZ community grew and experts began to gather and exchange information, educational programs where initiated in the Soviet Union. Major programs were developed in St. Petersburg, Kishinev, Novosibirsk, and Minsk, among other places. A group of TRIZ professionals became established at this time, and the rate of development of the methodology began to accelerate. Since Altshuller was beginning to reach advanced age, the period between 1980 and 1986 was a time of transition, where the focus went from the individual to the community.

With a variety of academies in place, and the new policy of *glasnost*, TRIZ began to come into the open. At first, the promotion was performed mostly inside Russia, but the increasing openness of the country in the late eighties and early nineties also allowed the rest of the world to be exposed to the methods of the Russian schools. This is also the time during which western companies and investors became very interested in the promise of TRIZ and began to form new companies and partnerships to lure Russian intellectuals out of their home country. These companies worked to further refine TRIZ and to adapt it to the needs of various technological clients.

Two of the main members of the Kishinev school of TRIZ, Zlotin and Zusman, continued their development of the field through Ideation International. The writings of these two scholars suggest that, in their view, the most valuable aspect of TRIZ is not in its problem solving capabilities, but in its predictions for the future [153]. By realizing a systematic way to solve design problems based on a nearly exhaustive search on historical analogy, the body of this analogy itself became available to the TRIZ community. Specific examples of technological progress have been condensed into general rules to describe future development.

TRIZ Methods

Classical TRIZ

It is worth noting that TRIZ is more about enumerating possibilities than deriving a small set of first principles from which the rest of the theory can be extracted. Many of the methods involve review a list of possibilities to jog the designer's mind until a solution comes into view. Thus, many of the methods will have a number corresponding to their informal titles, such as the "40 inventive principles" or the "39 inventive problems." A result of this is that there is no guarantee that the lists that are given in the TRIZ literature are exhaustive. In fact, it is likely that new possibilities will come into

play as science and the design process evolves. Those tools that do not revolve around enumeration may remind their users somewhat of brainstorming methods, such as the Contradiction Matrix or substance-field analysis.

One major goal of the traditional TRIZ techniques is to offset what was called "psychological inertia." In other words, a designer's expertise is a two-edged sword. On the one hand, it can lead to routine solutions in a shorter period of time than an inexperienced engineer would be capable. When it comes to creative or inventive problems, however, the experience base may cause a form of creative "tunnel vision" that prevents the engineer from seeing a clear, alternative path to the solution. Classical TRIZ methods involve transforming problems into a general form to help the practitioner see solutions before using experience to incorrectly reject them out of hand. A listing of TRIZ methods follows in this section.

The first of the classical TRIZ methods is contained within the 40 Inventive Principles [66]. This is often the first tool that is presented to a firm that is considering the use of TRIZ, since it is a straightforward problem-solving tool. The Inventive Principles are essentially an inventor's checklist, which operate along the lines of "have you tried this?" until the inventor hits upon an inspiration. The principles were derived from observations of how new inventions arose from their predecessors.

Closely connected to the Inventive Principles are the 39 Engineering Parameters, which lay out the types of problems that engineers are often concerned with. The parameters aim to capture the various metrics that an engineer attempts to improve in a product, such as the object's weight, reliability, efficiency and so forth. These are the types of parameters that are often involved in tradeoffs, where engineers and customers must work together to decide where benefits and penalties will be taken. Inventions that transcend these tradeoffs are typically successful.

Finding inventions that step beyond the current state-of-the-art in tradeoffs is what the Contradiction Matrix is for. This matrix is formed with headers on both the first

column and the first row that represent the engineering parameters [9]. One dimension of the matrix is put aside for the parameter that is to be improved, with the other dimension assigned to the parameter that is in direct conflict. For example, a machine may have problems increasing its output without becoming much more energy inefficient. The appropriate square of the Contradiction Matrix will then suggest a series of Inventive Principles to try and stimulate the designer to come up with a new invention. With the example of the productivity/loss of energy clash, a suggested solution is "Pneumatics and hydraulics." "Preliminary action" such as pre-staging or using self-contained cartridges is another suggested path to the solution.

Another useful tool in the classical chest is the Substance-Field Analysis [136]. This is a truly analytic method, rather than enumerative, and is not distantly removed from techniques of systems study such as functional decomposition. The idea is to break a problematic system down into components (substances) that either produce or receive effects (fields). While building the model, it is often discovered that a field is not sufficiently strong to achieve the desired outcome, or that the field has unwanted side effects. The Substance-Field Analysis can also be used to understand how to squeeze the maximal performance out of a system by more fully aligning the effects of its subsystems.

Substance-Field Analysis is best suited for problems in which the system has excesses or conflicting operations amongst its various subsystems. The generic form of the analysis is also useful in stifling preconceptions of the appropriate structure of a system while creative solutions for improvements are found. Finally, the method of analysis is useful for identifying where subsystems can be substituted, added or removed in order to improve the overall cost-benefit ratio of the system.

Tied to the Substance-Field Analysis is the final enumerative method of classical TRIZ. The Standard Solutions are a list of prescriptions for how to deal with the problems that are illuminated by the Substance-Field Analysis.

The culmination of the classical TRIZ methods is the realization that the various enumerations capture the wisdom of the full patent base. This implies that general patterns of technological evolution can be discerned from the patent base, traveling along paths suggested by the creative methods of TRIZ. The patterns of evolution are also enumerative, meaning that a new pattern could theoretically be discovered at any time. Altshuller himself enumerated eight patterns of evolution. Many more patterns have been established by those that followed in his research path.

The patterns of technological evolution form the element of TRIZ that is of greatest interest to the research to be undertaken in this proposal. A variety of technological forecasting methods have been proposed with TRIZ at their cores, including Guided Technology Evolution [73] and Directed Evolution [152].

Guided Technology Evolution (GTE)

The Guided Technology Evolution (GTE) technique is based strongly on the classical patterns of technological evolution. The goal of the method is to insert potential futures of the system, drawn from the patterns of evolution, into forecasting methods. This technique proceeds in four stages:

1. Analysis of the system's evolution
2. "Road mapping"
3. Problem formulation
4. Problem solving

The technique starts by a patent or technical paper search, which is used to understand the history of the system's development and to plot it on a typical S-curve. Altshuller's diagrams of invention level and numbers of inventions over time can also be used to understand where the system is in its evolutionary course. When the current state is understood, roads forward can be projected using the patterns of evolution. The lines of evolution will guide the analyst in seeing the future functional and structural changes

within the system. The ways forward will lead to the problem formulation, where engineering challenges can be identified. Finally, a set of projected solutions to these problems can be posited via the classical TRIZ tools.

This method of technology forecasting is less adapted funding profiles and parameter forecasting than it is for looking toward future designs and patents. In fact, one of the selling points for TRIZ technology forecasting is that it can show a critical path of invention and techniques to bring out a desired product. This path can then be used to understand what technical information needs to be protected so that the company undertaking a given development can maximize its competitive advantage. The idea of the "patent fence" has been promoted in more than one TRIZ article.

Directed Evolution (DE)

Directed Evolution is a more recent method, which has been formulated and promoted by Ideation International. In addition to incorporating more patterns of evolution than the classical eight, this method includes a variety of Ideation extensions to TRIZ. These extensions are not important for understanding the workings of Directed Evolution, but it is helpful to mention this

Directed Evolution (DE) is composed of five steps:

1. Collection of historical data
2. DE diagnostics
3. Synthesis of ideas
4. Decision making
5. Supporting the process of evolution

The historical collection phase of DE is more in-depth than that in GTE, and involves looking at several other factors in addition to the patent or system history. The functioning of the system itself is to be understood, including a breakdown of functions as well as favorable and harmful effects within the system. A history of the

contradictions and problems that have arisen within the technology are considered useful, along with an understanding of the environment and neighborhood of other systems that interact with the system of interest. Finally, external factors such as the market and organizations involved with a given product should be studied. For the TRIZ consultant, these goals are usually achieved through a series of questionnaires.

The DE diagnostics are referred to as the "heart of the DE process" in the DE manual [152]. The goal is to compare the predictions of the patterns of evolution to what has actually happened within the system to date. Dead-ends in evolution attempts, signs of psychological inertia in development, successful patterns of evolution undertaken and so forth can be used to predict the future state of a given technology. The current state of the system can also be analyzed through TRIZ to understand deficiencies, contradictions, unresolved harmful effects and other areas for improvement. Filling these holes is the likely work of future developers, and foreseen actions for doing so can be used to predict the system's future state.

Once a set of outstanding inventive problems for the system has been developed from the last two steps, it is time to look toward the future. The step of idea synthesis is meant to identify directions of technology improvement. This can involve a new market or function for the system to fulfill or can reduce the drawbacks inherent in using the system. At this point, a large number of possible paths forward should be formulated in a brainstorming process.

The fourth step is to begin making decisions and plans based on all the information gathered on a system to date. Evolutionary paths should be collected into compatible groups, with short-, middle-, and long-term goals laid out along each group of paths. This will enable action plans to be developed and provide for the necessary resources to be laid out.

Finally, the evolutionary process should be treated as a control problem. The evolutionary paths formulated in the previous steps are simply one set of expected

dynamics, which could be diverted by outside influences at any time. The key to supporting the evolutionary process is then to understand these outside influences and to have contingency plans prepared for when indicators show that one outside influence or another is in effect. This provides the "directed" part of the Directed Evolution name: a company or individual undertakes an active role in managing a system's technology down a pre-planned path.

Software and Added Lines of Evolution

A major thrust in the TRIZ field is to better package methodologies and to prepare users to practice them more rapidly. To this end, several TRIZ companies have written software [8], [17], [6], [18]. This software tends to operate like an electronic manual for TRIZ, with walkthroughs of proprietary questionnaires and guides to move through appropriate methods. Software packages can also include tools to build up substance-field models and run searches through a patent database in order to analyze a technology's history. The most advanced technology in TRIZ software is natural language analysis, which is turned toward understanding archived knowledge bases.

CREAX's Innovation Suite contains an expanded set of patterns of evolution that leads to more specific suggestions than the original eight. This software also has a set of examples to help the user understand what each of the patterns of evolution means. The original set of eight patterns has vague or obvious entries, such as "increasing ideality" or "technology follows a life cycle of birth, growth, maturity and decline." Specific laws such as "increasing dynamism and controllability" and "increasing complexity, followed by simplicity through integration" are more useful. The Innovation Suite expands on these patterns with practical entries such as "increasing dimension of volume," meaning that higher-order effects, such as three-dimensional flow are increasingly harnessed as a technology evolves. The software has 28 such patterns.

The importance of the new patterns of evolution in this context can be seen when

thinking about the source of technology evolution. Engineering is a discipline as much about practice as it is about theory, and so technology evolution is often a product of experience. When more patterns of evolution are laid out, they show the paths along which a designer should attempt to increase experience or background scientific knowledge. Taking the "increasing dimension of volume" line as an example for a turbine blade, the line shows that a turbine blade will reach its peak performance when it is designed as a fully optimized three-dimensional surface, rather than a flat paddle or twisted blade. In order to be able to optimize the blade, the full three-dimensional flow must be understood, and that the fluids science is sufficiently advanced to be rendered into practical analysis.

Ideation goes further than CREAX by claiming to have identified more than 400 separate Lines of Evolution. This may seem like a large number, but they are organized as special cases for a set of 12 general Patterns of Evolution. Several of these lines look outside the technological system, which also reduces the large number of possibilities.

Summary of TRIZ Material

Although few ideas were employed from the TRIZ body of work in the final method, they have been presented here for completeness. Since many of these ideas have developed independently from the ideas of scholars such as Basalla and Arthur, they are also valuable where they concur with the Western authors. If these two distinct lines of thinking drawn from the history of technology development (TRIZ from patents, Baslla from his studies of history) produce similar interpretations, then this provides some reinforcement for those ideas. Ideas in TRIZ that fall under this heading are the emphasis on scientific effects and the idea of even great inventions following common strategies and inspiration from other fields.

Also, the ideas embodied in the Lines of Evolution and evolutionary paths that are discussed in this material were considered as a way to forecast the performance of

components. While the guidance to look from two-dimensional to three-dimensional effects or from contact interactions to electrical interactions can be helpful to inventors, it is less helpful to someone attempting to predict the amount of improvement these actions will provide. It turns out that several other bodies of work on technology forecasting are more useful, as will be explained below.

Technology Forecasting and Roadmapping

Background

In the world of engineering, technological forecasting is vital to developing long-range plans. Conceptual designs are built entirely around the expected levels of performance of various sub-systems and technological solutions cast into given roles. If one technology is surging to replace the more traditional selection, it is important to understand the new technology's trajectory. If that trajectory is rising high enough to justify the risk of taking on something new, then the technology will have bought itself a place on the new design.

Technological forecasting is important not only to businesses, but also to global politics. Faulty intelligence can promote deleterious policy maneuvers and, worse yet, can ignite wars. Uncertainty in the estimation of weapons progress leads to potential security crises between nations. These crises open the door for arms races or even invasions. Thus, the difference between a high-quality, high-certainty estimate of the state of technological progress and one of lower quality can be counted in tens or even hundreds of thousands of lives. Of course, this line of logic assumes that current political leadership (and the voting public) is responsive to the results of complex scientific investigations.

The above scenario illustrates the main purpose of forecasting, which is to serve as the basis for future strategic decisions. By understanding the flow of progress with no

change in the way of doing business, decision makers can search for deficiencies. If, as in business, the progress of technology is found to be too slow, resources can be committed to speeding up or changing the direction of progress. If, as in global affairs, a technology is perceived to be progressing too rapidly, political actions can be applied to retard or divert technology development.

Forecasting for technology is based on similar ideas as forecasting in any other walk of life. The goal is to find some indicator in the present that can be used to understand the way that the future will unfold. Weather and extraction (as in oil or minerals) forecasting are two of the most advanced forms of forecasting, as they are now solidly based in physics. Atmospheric and fluid dynamics are used to forecast the weather using satellite and balloon observations around the country, while mineral extraction has geology and the history of mine construction to guide its forecasts. However, providing forecasts is a significantly more challenging endeavor for financial markets, since many of the factors that control a business' performance in the future are difficult to obtain or estimate. A staggering variety of technical indicators has been developed for forecasting stock prices from time series analysis [54], and there are as many rules of thumb as there are places on the bookstore shelf for celebrity stock brokers.

Technology forecasts aim to identify four elements [115]. The first element is to narrowly identify the actual technology whose future is to be forecast. This is not as intuitive as it may seem, since a technology may be defined in multiple ways. For example, "composite structures" can either be the "black metal" that slowly takes over an aircraft, or it can be a specific methodology, such as a specific autoclave-baked layup. The second element of a forecast is the time in which the technology attains a certain level of realization. The third element is the capability of the technology to perform its given function. Finally, the forecast contains a probability of its truth, which is defaulted to 100 percent, but should realistically reflect the level of the forecaster's confidence.

Martino [116] defines four major categories of technology forecast:

1. Extrapolation: the future of a time series is completely captured by its past, which needs only to be extended according to a descriptive law (trend, cycle, growth curve, etc.).
2. Leading indicators: the future of a time series is completely captured by the past of another time series, the leading indicator. A change in the leading indicator will be reflected in the following series (e.g., changes in the Producer Price Index typically lead to changes in the Consumer Price Index).
3. Causal models: the relevant variables and their linkages are known and can be described in mathematical equations (such as those for celestial mechanics).
4. Stochastic methods: these forecasts give odds on a given technological outcome rather than a deterministic prediction.

These four elements are used either alone or in combination in order to forecast the future of technologies.

Technology roadmapping is a different, but related, activity. It bears a similar relationship to technology forecasting as spiral development does to evolutionary acquisition. Roadmapping represents a series of decisions and plans that are made to move from one level of technological capability to another. Like spiral development, it is a method to develop a path from the starting point to the end. Roadmaps are more concerned with the details of progress than forecasting (at least in terms of resource allocation and required infrastructure), but roadmaps typically deal with a larger scope as well. Imbedded in many modern roadmaps are mappings between various levels of an enterprise, from its markets to its products to its manufacturing processes to its core technologies and its basic research.

A technology roadmap is described by Sandia National Laboratories [49] as a document that "identifies (for a set of product needs) the critical system requirements, the product and process performance targets, and the technology alternatives and milestones for meeting those targets. In effect, a technology roadmap identifies alternate technology

'roads' for meeting certain performance objectives … The roadmap identifies precise objectives and helps focus resources on the critical technologies that are needed to meet those objectives."

In other words, the process of developing a technology roadmap is to work backwards, beginning with an unsolvable problem and then lining up technologies to close the gap between now and the time of the problem's solution. This inherently means that technology forecasting will be used to ascertain when a given technology will be capable of what level of performance. Further, it is up to managers or the technologists to understand how many resources need to be allocated in order to bring a given technology to fruition.

Developing a technology forecast, and then using this information to develop a technology roadmap, forms the foundation for strategic planning.

History

1945 The Air Force commissions a team of experts, led by von Karman, to make predictions on the future of aeronautics technology, which is reported in *Toward New Horizons* [83].

1950's Technological advance understood to be major factor in Cold War, and RAND Corporation is tasked with making forecasts.

1964 *Report on a Long-Range Forecasting Study* [82] is published, which describes the Delphi technique as developed by the RAND Corporation. This report contained a combined Delphi forecast from a variety of famous intellectuals, including Isaac Asimov, Arthur Clarke, Peter Goldmark and William Pickering.

1968 Floyd develops a causal model of advance by relating a technology's capability growth to the effort applied by researchers.

1968 (Edwin)Mansfield produces a techno-economic model for the rate of technology adoption, based on profitability and cost of deploying a technology, which is handicapped by an "innovation coefficient" for a given industry [112].

1971 Fisher and Pry develop a variant of the logistic growth ("S") curve to apply to the substitution of new technology for old [74].

1972 Martino and Conver show a relationship between maximal steam power plant installation size and total installed steam power generation.

1972 Kane develops a simulation to model cross-impacts (one technology indirectly advancing another, or social effects advancing technological progress) called KSIM [94].

1973 Blackman et. al extend Mansfield's techno-economic model for the rate of technology adoption in a variety of industries, using company research and development expenditures rather than historical data [42].

1979 Murthy publishes a stochastic method for producing technology forecasts.

1980's Motorola develops first technology roadmapping procedures.

1985 Foster applies the growth curve to technology metrics [76] rather than substitution.

1992 First Semiconductor Industry Association technology roadmap is produced.

Methods

Expert Survey

The expert survey is the oldest form of forecasting, and has likely been used since the days of the Pharaohs. Although it has become more structured in the modern age, with groups of experts answering structured questions, the core of this type of forecast is essentially the same. An expert with a proven record in the relevant field is asked for his or her advice on the future; and on what trends are foreseen within the field of expertise.

In wider surveys, multiple experts are gathered either to give their predictions individually or to cooperate in order to generate a more developed forecast.

The expert survey is a popular form of forecasting, and has been utilized in a variety of technology studies and roadmapping exercises. There are also a number of compound technology and strategic methodologies that have data from expert surveys at their core [129], [119], [38], [98].

Delphi

The Delphi procedure was developed to gather the thoughts of experts while mitigating the problems with group dynamics and individual psychology that serve to color a given forecast. Anonymity avoids harmful group effects such as ad hominem criticisms. Feedback is used to challenge and correct the statements of experts with factors that may not have been considered earlier. Finally, statistical results are used to show the full range of opinions and give the decision-maker an understanding of the level (or lack) of consensus in the forecasting group.

Delphi achieves the above goals of anonymity, iteration and probability estimation through multiple rounds of questionnaires. The first round is completely unstructured, meant to elicit a series of expected events and trends from the various experts. These events will be combined into a common set of expected events, and will be used to develop a second questionnaire. The experts will be asked to forecast an approximate timeline for the events that have been written into the questionnaires. Statistics of these responses will then be given to the experts to revise their estimates in the third round of questionnaires. If an expert's revised opinion falls too far away from the majority of the group, that expert will be asked to provide the reasons for his or her confidence in standing apart from the group. The final round of surveys is used for another forecast revision, and comments are solicited from the entire group. Moderators can then consolidate all of these data in order to come up with a forecast.

The Delphi method has been used when a large number of experts are available to provide answers and guidance. Many national research plans and strategies have been developed by this method, both in Japan and in Europe.

Growth Curves

Growth curves represent the manner in which most technologies develop. It has been observed that technologies start their lives in a slow advance, as the science is slowly worked out and only a handful of researchers are participating in its development. As a wider base of knowledge on the technology is developed, it becomes easier to see the fastest ways to advance it. As the technology matures, physical limits are realized and the growth rate again begins to slow down.

Growth curves can be implemented in one of two ways. The simpler, but more error-prone way, is to simply fit all of the parameters of the appropriate formula (given as some kind of sigmoid function like the hyperbolic tangent) to the levels of performance seen in a given technology so far. The second method is to use the laws of physics in order to predict the theoretical maximal performance of a technology, and use this theoretical limit as the value for the appropriate modeling parameter.

While a relatively simple form of forecasting (provided the requisite data are available), growth curves have been shown to accurately predict the growth of several technologies [77].

State-of-the-Art Curves

Traditional growth curve methods were most useful for technologies that only had one technical goal. For example, Moore's law is quite powerful in predicting the number of transistors in a given microchip. The driving interest of the microchip industry has been to shrink the features of a microchip, dealing with issues in manufacture, heat dissipation, leaking current and so on.

Martino developed a simple model for incorporating multiple technology attributes changing over time [114], given below as a hyperplane.

$$t = \sum_i K_i P_i$$

where t is the predicted time of introduction for a technology with the characteristic values P_i, and K_i are weightings to be regressed.

The test case for this model was a series of jet fighters produced from 1944 to 1982. He compared this model to a composite score that was provided over time, and showed a preference for the ability to trade future characteristics off against each other. In addition, he found that the multi-dimensional model predicted the year of first flight better than the scoring model. In addition, the recommendation was made to minimize residuals rather than squared errors in order to avoid being biased by extremes in the data.

Earlier, Dodson had performed curve fits on rocket data [64] using a similar equation, which is given below. Also, rather than treating the residual between this fit and the actual data as purely a problem of empirical data, Dodson suggested that this represented the leading or lagging nature of a given technological release. Similarly, future developments could be judged in their aggressiveness by their location relative to the results of this model.

$$t = \sum_i \left(\frac{X_i}{a_i}\right)^2$$

In this case, t is the expected year of introduction, the values X_i represent the technology characteristic values, and a_i are weightings to be regressed to the data to be fit.

Danner [57] extended this work by developing a set of transformations that resembled standard growth curves in that they had asymptotic limits with respect to time. In his paper, he posited that even “single-dimension” curves often had assumptions about the prioritization of requirements and constraints embedded within them as a technology

advanced. He cites the example of turbine blades, which must withstand a given temperature, but have implicit in the definition of "withstand" a description of strength and fatigue performance. Thus, when an S-curve is developed even for a single metric, there are buried assumptions about the development path of other metrics within the same technology.

To solve this problem, Danner suggested the use of technology frontiers to capture the trade between multiple metrics. This frontier would then progress through time as investment is funneled to changing its shape. This can be seen in Figure 5. Interestingly, this is a similar viewpoint to that of TRIZ, which states that the chief contribution of technology's advance is to redefine the trades between different engineering parameters [144].

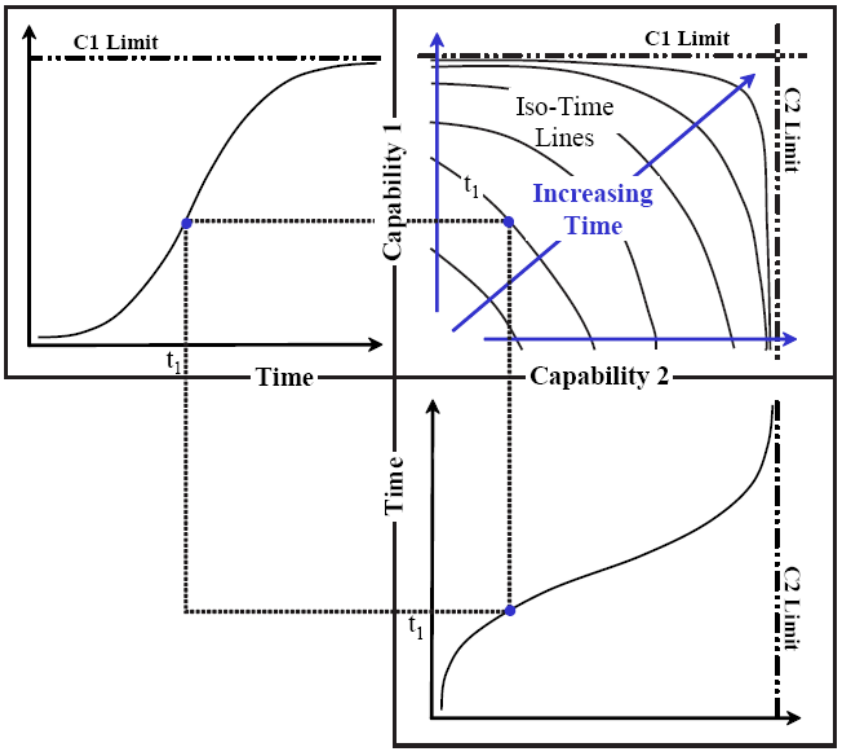


Figure 5: Relationship between S-curves and technology frontiers [57]

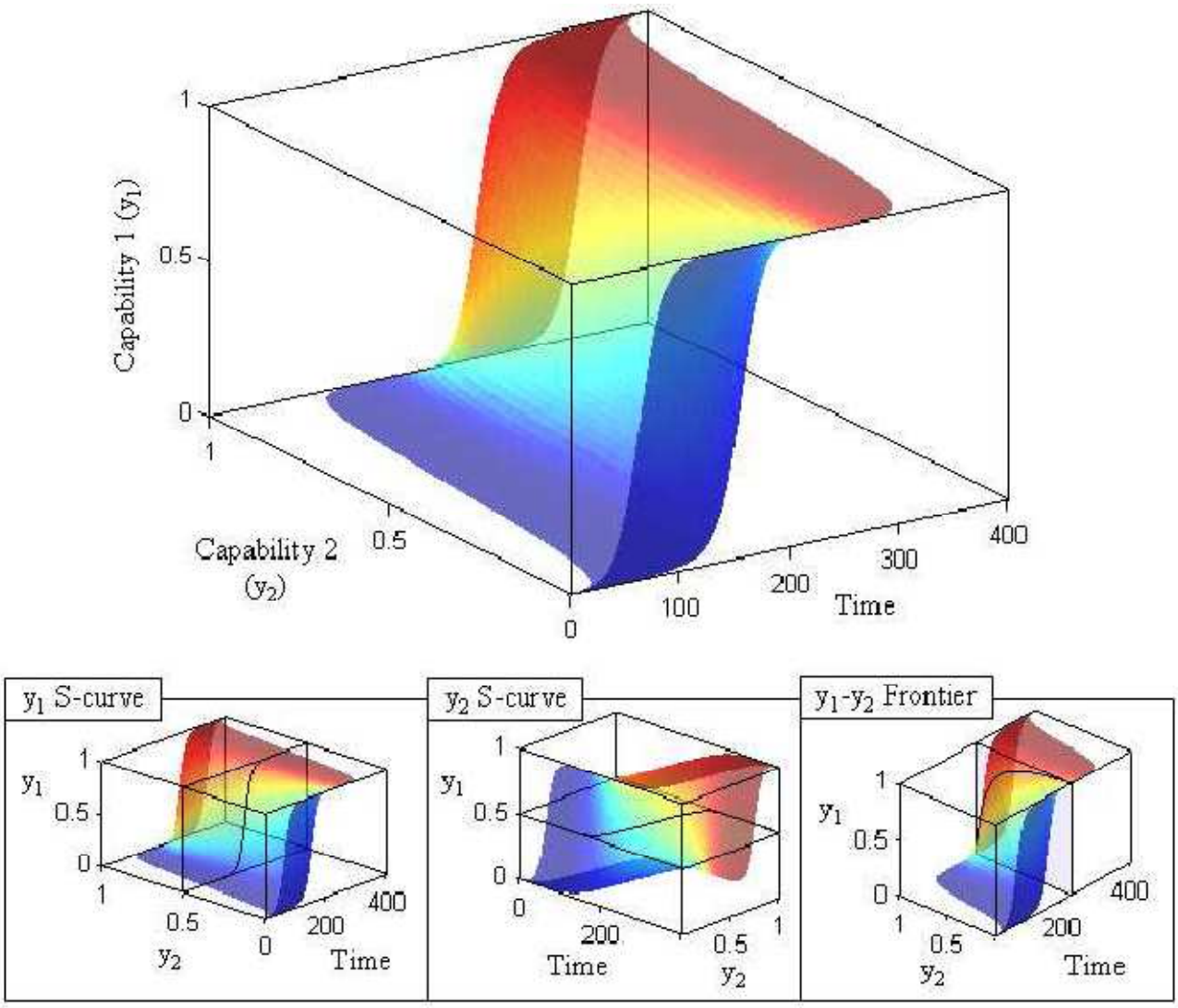


Figure 6. Visualization of multidimensional S-curve [57]

Danner's formulation relied on developing transformations on the technology variables and then performing a simple linear fit. Multiple forms are suggested throughout his thesis, with the most frequently mentioned form as the one below:

$$X_i = ln\left(\frac{L_i - y_i}{y_i - y_{0,i}}\right)$$

where L is the ultimate limit of a technological characteristic, y_0 is the starting point for that characteristic, and y is the current value of the characteristic.

Those transformed variables are then used in a linear least squares fit against time

of introduction with weights β_i in the following form:

$$t = \beta_0 + \sum \beta_i X_i$$

Once this is completed, a multidimensional surface of time and technology characteristics has been created, and can be navigated to project and trade multiple characteristics into the future. A diagram of this surface is shown in Figure 6.

Technology Forecasting through Data Envelopment Analysis (TFDEA)

TFDEA is in adaptation of a business planning technique called data envelopment analysis. The model was originally applied to industrial plants, which can produce varying levels of outputs for different levels of inputs. DEA identifies plants as either efficient or not, where efficient plants generate the most outputs of a preferred type for the least inputs of the set of plants under consideration. The relative level of efficiency of a plant is derived in DEA by a linear programming problem:

$$\max\ \theta_k = \frac{\sum_i \mu_i y_{r,k}}{\sum_i \nu_i x_{i,k}}$$

$$\text{s.t. } \frac{\sum_{r=1}^{s} \mu_r y_{r,j}}{\sum_{i=1}^{m} \nu_i x_{i,j}} \leq 1 \qquad \forall j \in \{1,2,\dots n\}$$

$$\mu_r, \nu_i > 0$$

In the above equations, θ_k represents the relative efficiency of DMU k, where 1.0 is efficient. The number of DMUs is represented by n, s is the number of outputs, m is the number of inputs, x represents an input and y represents an output. The solution to the above problem will give DMU scores from 0 to 1.0.

Transforming the above into a technology forecasting technique involves generating new DMU scores by projecting forward in time, in this case to account for the reduction of input required to deliver the same output. The projected DMUs form the basis for a new frontier, which can be used to project future requirements into an

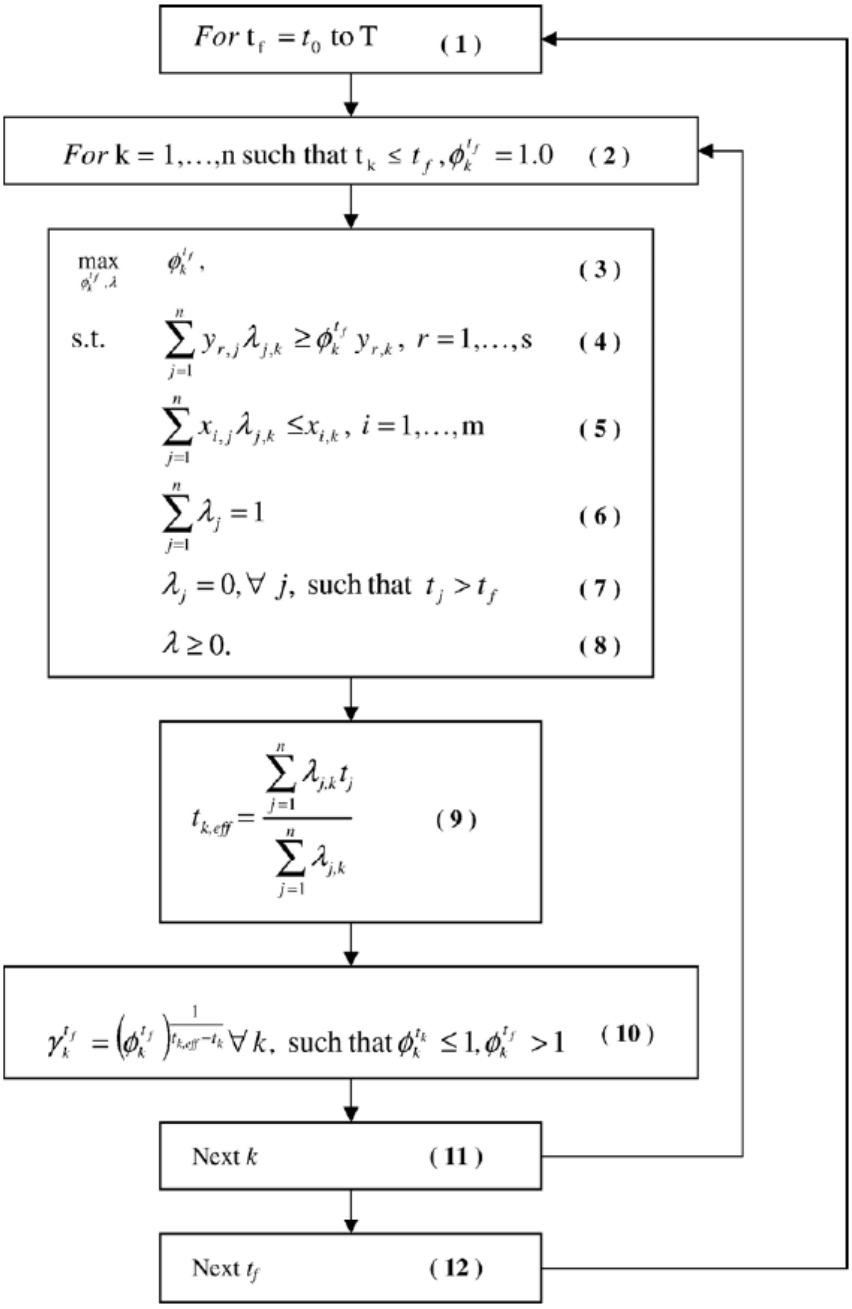


Figure 7. TFDEA future projection flowchart [92]

expected time of release. This projection represents a constant rate of change, as given by the following equation.

$$\beta_k^{(t_k+1)} = \frac{\left(\theta^{(t_k+1)}\right)}{\left(\theta_k^{t_k}\right)}$$

TFDEA was developed and explained in a thesis by Inman [91]. The full flowchart of the method can be seen in Figure 7. The flowchart shows a loop taken over all of the time periods of interest from t_0 to t_f. The loop also progressively changes the variable t_f, meaning that T is the ultimate end of the procedure, and that t_f is simply the period to which technology data all other periods are being projected. The next loop iterates over every DMU (technology data point) in the database before that time. The linear problem to find efficiency at every time is then solved, and that solution is used to develop a rate of change from the past to the present time.

In addition to several example problems (microchips, servers, disk drives), Inman compared his method with that of Martino's paper on fighter jet years of introduction [92]. He found a better match for his method in terms of mean errors between predicted and actual years of introduction.

Correlation Forecasting

Correlation forecasting relies on indicators related to a given technology to understand its rate of progress. One form of correlation is economic. For example, the installed base of consumer products, such as automobiles, telephones and televisions within a country can be related to the per capita gross national product for that nation. Of course, this correlation ignores social factors and generation skipping in technology.

Correlation forecasting works best when there is a metric that can capture the total investment in a technology, such as installed base. Wind power cost per watt would be a good metric to correlate with installed base, since cost is the chief driver of wind power acceptance and is driven by experience effects.

Lead-Lag Forecasting

Lead-lag forecasting is best applied to situations of technology transfer between various products, or from the laboratory to the real world. For example, a study [84] of aerospace applications of advanced composites showed a roughly 10-year lag between experimental demonstrations of composites and their adaptation to operational aircraft. This forecast has held up fairly well, properly predicting the composite weight fraction of the F-14, F-15, F-18, AV-8B and the F-22.

Anytime there is a hierarchy of applications, lead-lag forecasting can be helpful, so long as this hierarchy remains. Typically the order of introduction is to start with performance-driven systems and slowly move into more cost-sensitive systems.

Causal Models

The difference between causal models and other forms of technological forecasting is the same difference between conceptual design by physics-based tools and design by historical analogy. Historical analogies can be quite useful, especially since only data and a set of regression tools are necessary to acquire the basis for decision-making. However, there are three major shortcomings of data-based forecasts, as identified by Martino [115]:

1. They are unable to provide a warming that there has been a significant change in the conditions that produced the past behavior and that, therefore, this behavior will not continue.
2. They are unable to predict the future when the forecaster knows that an important factor affecting technological change will be altered.
3. They are unable to give policy guidance about which factors should be changed and by how much to produce a desired technological change.

Causal models attempt to model the technology development process itself. Simple models relate the level of effort applied to technological advance to its actual rate

of advance. More advanced models fold in the effects of economics, such as companies actually acquiring and producing a given technology, in order to predict advances. The most complex models would attempt to incorporate every outside influence to a given technology's development. An advanced model may use artificial intelligence to emulate the rate at which human reasoning can innovate modifications to the existing technological base.

One interesting causal model deals with multiple competing technologies, where one may replace another. Although the Lotka-Volterra equations are used to describe competing species in biology, they can also be applied in the technology realm if one considers technologies as entities competing for the right to exist. An interesting example for which these equations fit (up to the date of publication) was an analysis on energy source substitution between wood, oil, gas and coal [113].

Cross-impact Simulation

Cross-impact models recognize that technologies do not develop independently of the outside world. While the data-based methods of technology forecasting implicitly capture all of the dynamics of the research and development process, causal models must include these dynamics explicitly. This may make causal models more difficult to use properly, but this also accounts for their power, since they can make predictions based on changing external circumstances.

A cross-impact model can include impacts from the progress of other technologies (due to a common scientific basis), social, regulatory or economic factors. For example, a call for major carbon dioxide emissions reductions would cause an infusion of investment to non-combustion automobile technologies. Having these cross-impacts allows forecasters to posit scenarios, the indicators of which can be used to understand when a given technological development may accelerate or decelerate.

Motorola Roadmapping Process

A “canonical” technology roadmapping process that is often cited in the literature is due to Motorola [151], published in 1987. Two different types of roadmap are described. The first, the emerging technology roadmap, is only mentioned briefly in this paper. It is dedicated to understanding the company’s mastery of a new technology, relative to its competitors, and establishing whether it is important to commit resources to strengthen this mastery. The second type is the focus of the paper, and is called the Product Technology Roadmap. The roadmap is described as “a compilation of documents that provides a comprehensive description of the product line – past, present and future – of a division or operating group.” With this information as a foundation, plans can be laid for taking maximal advantage of the future.

The technology roadmap as prescribed by Motorola at this time has eight sections, or products. A consolidated list is pictured in Figure 8. The first section is the description of the business, which lays out business mission, sales, previous experience with the product, and who the current competition is. This section is one of the most intensive, since it defines the current state of the industry and often has the best data associated with it, since it is based mostly on accounting sheets and information from recent meetings. Another large piece of work, the technology forecast, forms the second section of the roadmap. Future plans for the product from the first section and the technology forecast are combined in the third section to form the roadmap matrix. This matrix is akin to a Morphological Matrix, except that expected changes in technology are laid out chronologically on the horizontal instead of current alternatives. This is an element of the roadmapping process that is of pertinence to this dissertation. The matrix shown in Figure 9 shows the result of expert opinion in forecast which technologies will become dominant in given times.

1. Description of Business:
- Mission
- Strategies
- Sales History & Forecast
- Product Plan & Lifecycle
- Experience curve data
- Competition

2. Technology Forecast

3. Technology Roadmap Matrix

4. Quality Considerations

5. Quality Considerations

6. Patent Portfolio

7. Product Descriptions, Status Reports, & Summary Charts
- Objective
- Key Personnel
- Budget
- Staffing

8. Minority Report

Figure 8. Products of Motorola roadmapping process

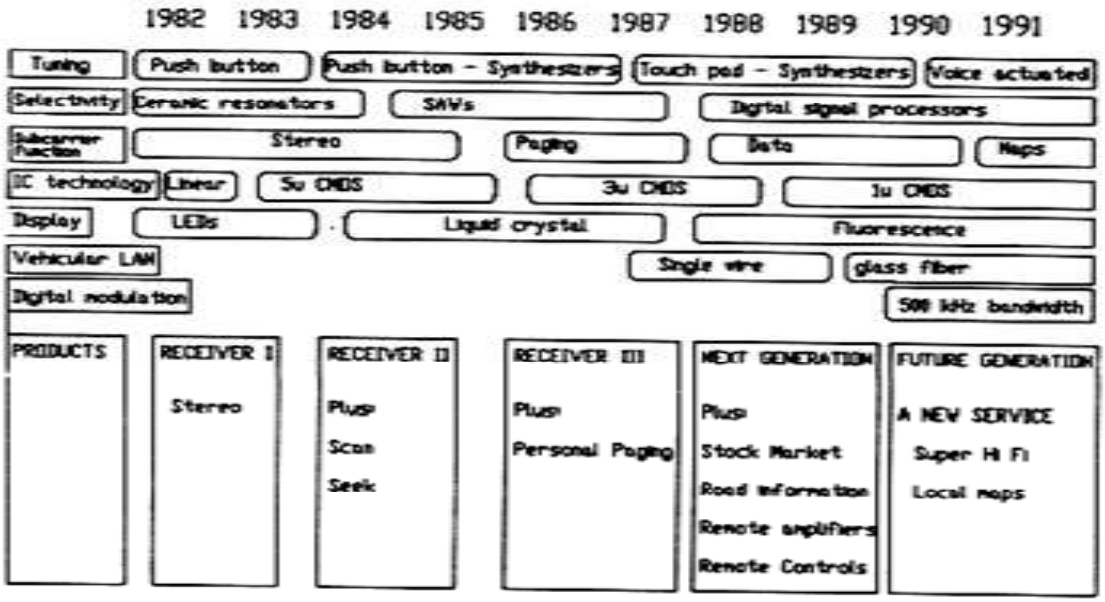


Figure 9. Technology roadmap matrix [151]

Fourth to be considered is the impact of new technologies on product quality. The fifth section is where resources are mapped to product plans. A patent portfolio is constructed for the sixth section. A series of summary charts and product descriptions with objectives, key personnel and funding levels, which is meant to be maintained and updated, forms the seventh section. Finally, a minority report can be amended to the report if there are technologies that might be considered but not strongly advocated in the report.

Sandia Roadmapping Process

One process that many other papers cross-reference in the literature is a method [49] put forward by authors at Sandia National Laboratories. This process is described as proceeding in three phases: preliminary activity, development of the technology roadmap, and follow-up activity. These workflow within these phases is illustrated in Figure 10.

The goal of the preliminary activity is to build consensus and to ensure that the various stakeholders that may benefit from a roadmap get at least some of their needs satisfied. A series of essential conditions should be fulfilled before proceeding: the roadmap must be needed, the effort is worthy of and should garner involvement of multiple contributor groups in the organization, and the scope of the roadmapping is needs-based and well-defined. The preliminary work must also establish the appropriate leadership, just like any other project that is undertaken. Finally, the scope of needs (whether market-, product-, or effects-based) and time should be delineated and agreed upon.

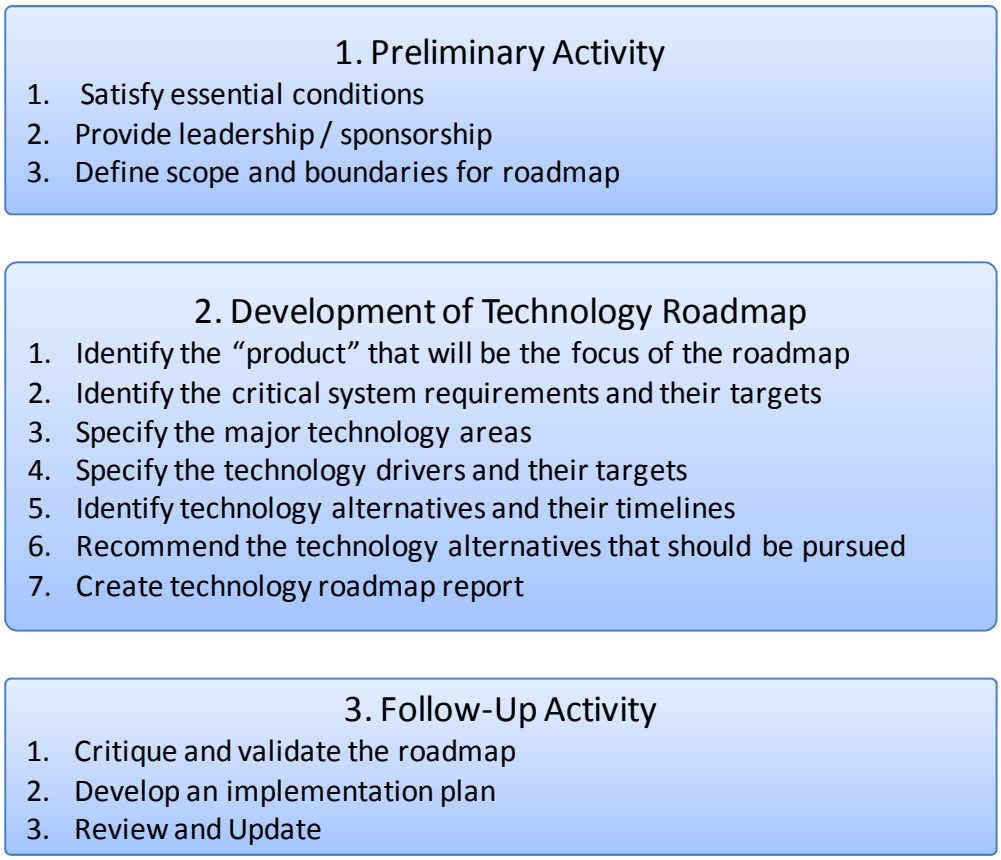


Figure 10. Steps in Sandia roadmapping process

The development of the roadmap is broken down into seven steps. The first step is to identify the product or products that will be the focus of the roadmap, such as an energy-efficient vehicle. Scenarios can be devised to identify how the utility of products varies based on conditions in the outside world. Critical system requirements and targets for improved values are derived in the second step. Areas for major technology investment to achieve these improved values are specified in the next step. The fourth step is to break these technological areas down into drivers and targets. With the energy-efficient vehicle example, a technology driver for materials may be vehicle weight or acceptable engine temperature. With the technology parameters defined, the next step is to identify a set of promising technology alternatives. The technologies are then evaluated, and the best choice (or portfolio) is recommended. Finally, all of these results are collected together and used to draft a roadmap report.

The follow-up activities are meant to develop a second round of buy-in and consensus building for the roadmap. Implementation involves a larger group than the original planning in many cases, so these added participants must buy in to the plan. This begins with a review and critique of the work, where it is ensured that the proposed technology portfolio will succeed in meeting targets and that useful alternatives were not bypassed. Once the larger group improves the roadmap, workshops are recommended to decide on how to implement the plans. Finally, the roadmap should be reviewed and updated regularly, as new, applicable information surfaces.

Science Roadmaps

Science roadmaps extend the ideas of technology roadmapping to the domain of basic research. Like technology roadmapping, science roadmapping is meant to allocate finite resources wisely to potential research activities. It is important to also realize that scientific research is more unpredictable than technology research. A given experiment can raise new questions and call for new experiments to be inserted into the plan. Science roadmaps can also be seen in an oblique way as also being technology roadmaps, since the development of scientific apparatus is key to experimentation. This can be taken in the converse direction as well, meaning that the experimental apparatus for a new scientific phenomenon can also serve as a very early prototype for exploiting it.

Like technology roadmaps, science roadmaps begin with a need that must be addressed by planning. In the scientific realm, needs are typically driven by gaps in current understanding, as revealed either by unexplained phenomena or by computer simulations that have proven incapable of describing reality in some way. This is also true for the engineering sciences, although the questions typically arise due to a lack of understanding of how to cause one effect to precipitate another, such as coaxing a given material into superconductivity.

While technology roadmaps tend to focus on the technologies themselves as

programs, science roadmaps more often point to facilities or experimental apparatuses. Excellent examples are NASA roadmaps for space and planetary science with spacecraft to be deployed [13], [5]. The particle physics community is another case where experiments and facility improvement are closely tied with one another [141].

Strategic Prioritization and Planning (SP2)

The Strategic Prioritization and Planning method [99] is a new development in strategic planning. While it is more general than a technology roadmapping technique, it can certainly be employed for this purpose. SP2 is a method to systematically map different levels of problem abstraction from one to another. Through this method, such disparate factors as the national mission of NASA and technology programs could be related to one another.

The core of SP2 is the use of computer technology and planning matrices that relate one level of abstraction to another. The matrices are the natural descendants of quality methods such as the quality functional deployment (QFD). In more abstract levels of a given problem, these relationships will be highly qualitative, such as the mapping between national needs and departments of research efforts. Toward the lower levels, such as mapping technology to subsystem improvements, a quantitative mapping is possible.

Since SP2 is based on computer technology, it is possible to develop a database that can be used to work problems either top-down or bottom-up. Funding levels can then be mapped to top-level requirements and vice versa. When the mapping is between system requirements and technologies, this mapping can typically be quantitative by utilizing the methods of TIES. However, this also pre-supposes a set of technologies to be researched.

Finally, SP2 is able to develop plans with a greater traceability than previous workshop and survey-based methods. While SP2 also uses expert input, it organizes the

results in a traceable and transparent way so that assumptions can be examined and challenged. It is also rapid enough to allow the drawing up of a final roadmap (or multiple roadmap scenarios) to be conducted within an interacting conference rather than among a small group of leaders.

Summary of Technology Forecasting Material

Technology forecasting and roadmapping techniques come in many variations, but most are either based on historical data extrapolations or expert opinion. Refinements are available in removing bias, examining the effects and consequences of results, extrapolating from one application to another, and accounting for multiple technology characteristics simultaneously. Roadmapping techniques take forecasting inputs and develop them into integrated, strategic plans over multiple years. In addition to being static documents, there are techniques being developed to make them far more interactive. They also have served as effective guides on what to expect in the future, and have many merits to consider moving forward in this research.

The key takeaway from this section is that there are a large number of techniques that can be used to forecast the future performance of components. The forecasts of future component performance will be a necessary foundation for evaluating the performance potential of architectures composed of them.

Combinatorial Specifications

Although this is not directly related to technology *per se*, this section is meant to quickly orient the reader to some important aspects of the kinds of problems posed by selecting a set of components and connections with which to build up an architecture. The problems and techniques described here will be utilized in the next chapter as a solution to the problems discussed earlier with architectures is formulated.

Any time a new aerospace system is conceived, there is a large number of choices

to be made to arrive at a configuration. In the world of traditional aircraft, there are choices such whether to employ a high, middle, or low wing. Engine type, engine location, empennage configuration, fuselage design, and many other selections must be made. The selection space often becomes larger if there is a radically new configuration to be considered, as there may not be a well-defined set of subsystem interfaces but rather a set of functions that must be fulfilled by a handful of subsystems within the configuration.

A systematic way for laying out a set of possible combinations of choices was laid out by Zwicky [154], which is called the Morphological Matrix. In this formulation, he described a series of bins or drawers that could be filled with different options. In the traditional formulation, the subsystem or role (e.g., wing location or engine type) is given a row, and the various choices are given columns. The user of this matrix chooses one column in each row, and a configuration is considered the set of choices for each row.

There are two major extensions to the Morphological Matrix to highlight here. The first is the use of an auxiliary "compatibility" matrix [98] in order to capture knowledge of combinations that do not physically work together. The second is the development of a computerized version of the matrix [70] called the Interactive Reconfigurable Matrix of Alternatives (IRMA). This matrix is utilized in brainstorming sessions and discussions to walk through the various options of interest to the design team. Further, it can be used to filter out alternatives that are not sufficiently technologically mature or fail to meet either criteria for consideration.

Another path that is considered in this work is the body of work in mathematics known as graph theory. Graph theory is concerned greatly with both combinatorial problems and network problems. Many of the combinatorial problems that graph theory has been applied to have other solutions, but often the graph supplies a viewpoint that can either simplify or describe more intuitively the solution.

The basic elements of graph theory are vertices, which are depicted as points or

unfilled circles in a graph, and edges, which are depicted as lines. Vertices are usually considered as the main entities of consideration, such as valves, people, cities, or items. Edges represent either a physical or conceptual connection, and also typically hold the key to a given solution or proof on a graph problem. Edges may be directed (can be traversed only from start to finish) or undirected, and may have quantities or colors attached to them.

One common graph theory problem is to examine the connectivity of networks, such as how many redundant pathways serviced a given vertex or many vertices are crossed when trying to connect two points on the graph [51]. Social networks can also be analyzed, such the average number of "friends" a given vertex has or the degree to which vertices cluster into mutually exclusive "cliques." [150] Finally, there are a wide array of problems that deal with the flow of a given quantity through a graph, and is the subject of multiple theorems [61]. In each of these problems, the properties of the set of edges is typically the focal point.

Another common problem is to find a given route through a network. The traveling salesman problem, which is to find the least expensive itinerary through a series of vertices returning to the same point (a tour), is one of the most famous. Finding shortest paths through a graph is another common problem.

The body of combinatorial problems to which graph theory has been applied is also interesting. There are a variety of matching problems, an example of which is the problem of sorting through a number of prospective employees and seeing if the set of jobs for them can be filled. Another variant is to assign scores to how well the jobs are filled, and to attempt to maximize the overall fit of employee or task. Although edge information is useful to solve the problem, the focus is on how the problem affects the vertices.

The job assignment problem can be easily seen to be applied to configuration problem, namely assigning a given subsystem to the role in which it can be used in

system. As will be seen in the formulation chapter, there are also a number of more sophisticated ways to represent a configuration using graphs. However, the main theme of vertex as object and edge as connectivity information will be preserved.

Summary of Chapter

Each section of reading has been summarized separately, but it is worth a moment to consider how these sections are to be synthesized into a single method. The reading on evolutionary views of technology are best applied at the architectural level. The fact that technological components are often repurposed into new architectures and uses implies that a formulation of architecture that is independent from strict assignments of functionality may be desirable.

The technological forecasting portion of the chapter lays out a rather large toolbox of potential techniques that will have to be narrowed down in order to select one that will be used to account for progress in component performance. Once this selection has been made, the appropriate technique will be added to the method.

Finally, a short background on a pair of approaches to combinatorial problems has been introduced. This information will be expanded upon in the formulation section as it is applied to the main problem of this dissertation.

CHAPTER 3

RESEARCH POINT OF DEPARTURE

This chapter develops a research plan based on the literature survey of the previous chapters and some philosophical considerations. From these musings, a series of research questions and hypotheses will be put forth for the rest of the thesis to answer.

There is a great deal of literature available on the management and forecasting of technology, as seen from chapter two. Technology has been understood to be an extremely important factor in the nation's success, especially World War II. The military was the among the first to invest large amounts of resources into understanding the future of technology, and has multiple studies and techniques to show for this investment. The techniques of forecasting soon entered the realm of commercial competition as well.

Evolutionary acquisition provides a logical framework for bringing major technology improvements over a long period of time and with highly uncertain requirements. In doing so, however, it raises a new problem. How does one construct a technological forecast for an effort that will, almost by definition, change its direction and reallocate priorities? This is where a process-based model is needed, to play "what if" games that concern both the technology suite a given system is founded upon and the release strategy of the system itself. The technology and release schedule problems are coupled, as it is the technology that determines when a given requirement can be satisfied to a desired level.

The state of the art in evolutionary acquisition is shown in Figure 11, which presents the four pillars of interest in this paper and their interchanges. In this diagram, the green arrows indicate quantitative methods, orange arrows indicate qualitative methods and red arrows indicate ad hoc or overstretched methods. The methods highlighted in this chart are by no means an exhaustive list; they simply represent the

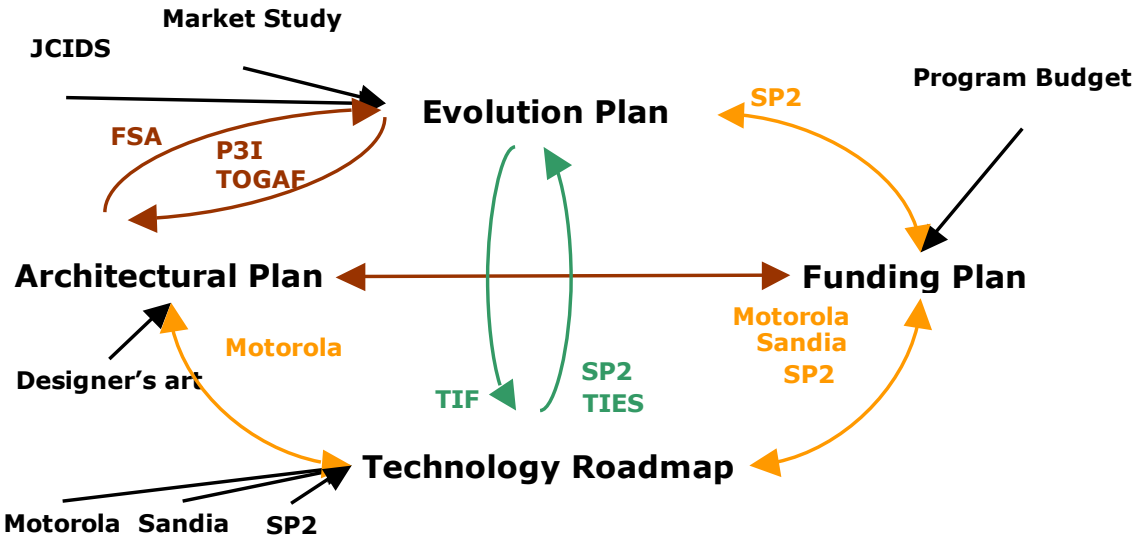


Figure 11: Methods to go between pillars in evolutionary acquisition planning

most applicable to interchanges between the different pillars. There are several ways to develop the evolution, architecture, technology and funding plans, and each of these enable interchange between them to varying degrees. However, many of these approaches are either ad hoc or qualitative, being based on expert opinion or chief designer's art.

A good place to start is the development of time-phased requirements. The military now intends to develop them primarily through war college activities and analyses that are formally encapsulated in the JCIDS process. In the commercial world, market analyses, focus groups and user surveys are used to see where the domains of growing demand lie. These are sources of requirements prioritization and desires, which form the technology pull. Meanwhile, technologists contribute a view of the possible and which technology attributes are easiest to improve. Desires and expectations of the possible are meant to balance requirements during the Functional Solutions Analysis (FSA) portion of the JCIDS process.

The design of architecture is still more art than science, although the advance of

computer tools are enabling this art to be practiced in an increasingly disciplined way. Many ways of defining an architecture have been proposed, but again, these are tools for development rather than fully constructed methodologies. It is worth considering an analogy to engineering design in general; it is typically far easier to have processes to specify and analyze than it is to make a process for design. Also, this is the state of architecture design for static cases. Evolutionary acquisition requires a plan for evolving an architecture from an initial state, which further complicates this situation.

The interactions between these aspects must be considered. The connection between time-phased requirements and architecture evolution are ad hoc. The methods currently used to plan architectural changes are the development of technical standards and change management. Change management is typically a reactive process, preparing an entity to cope with change rather than planning it in advance. Technical standards are best at defining external interfaces, but not what happens within a given system. Development efforts undertaken in pre-planned product improvement (P3I) or other incremental approaches will attempt to reserve resources (mass, power, etc.) for potential add-ins, flowing time-phased requirements into architectural evolution in a somewhat orderly way. In the other direction, analysis of alternatives and solutions (e.g., during FSA) will inform the link between architecture changes and time-phased requirements.

Technology roadmaps implicitly include plans on funding and architectural change. This inclusion is typically qualitative on the architectural side, as the forecasts of architectural change are driven by expert opinion rather than explicit design. On the funding side, this transition may be qualitative or quantitative, depending on the amount of detailed analysis performed in developing the road map.

The best understood link is that between requirements and technology. The success of TIES [98] and TIF [101] is that they can quantitatively map between requirements and technology. These two methods will be examined closely for lessons that can be applied to applying more quantitative techniques to the interconnections

between all four pillars.

The tie between funding and time-phased requirements exists, but it is more implicit than that between technology and funding. In fact, current methods typically use the technology-funding link as the basis for connection. This link is typically developed, and then the technology plans are mapped to the time-phased requirements they aim to achieve. Strategic Prioritization and Planning (SP2), being based on computing methods, can execute this implicit link rapidly enough that it forms a de facto link between time-phased requirements and funding.

The chief gap that this research was formulated to fill was the lack of ability to quantitatively map between time-phased requirements and architectural shifts. TIES fills some of this gap, but it requires a defined technology portfolio to work with as well as either a static architecture or a number of well-defined alternative architectures. In this case, what is desired is the ability to link time-phased requirements and architectural change in a much more general sense. To this end, the hypotheses that appear later in this chapter were formulated.

The rest of this section will be organized in the following way. First, a major *research question* will be posed. This may be answered by simple conjecture based on the literature survey. If further investigation is required, a *hypothesis* to answer the question will be offered and demand original work and testing. Finally, *falsifiability conditions* will be derived from these hypotheses as potential methods to demonstrate their falsehood. If a falsifiability condition can be shown to be true, this puts the hypothesis in jeopardy. If the condition is not met through testing, it provides credence to the validity of the hypothesis. Falsifiability in this context is softened to rate a hypothesis as stronger or weaker relative to experimental outcome, rather than an attempt at strict elimination.

Research Question #1 *Is there a naturalistic explanation of technological progress based on cause and effect?*

The various biological, complexity, and economics insights have the potential to transform technology planning models from data-based constructs to process-based ones. The many authors each appear to be identifying at least one important point about technology and its evolution. A synthesis of these points would be quite fruitful in developing models.

Arthur's insight [34] as to the nature of technology is a most useful one – namely that intentional inventors are out to pair a known effect with a given problem. The matching game can arise from a driven search (needing a solution to a problem) or from the observation of an effect leading to its application. This is the advantage human intention has over blind evolution. Further, Arthur describes how the main function will often require supporting functions to construct the hierarchy of a technology, which mirrors how complicated systems are actually designed.

TRIZ has also brought insights, especially in its base of patents showing not only technological progress, but the typical impact of individual inventions as a given technology matures. Possibly one of the greater insights of TRIZ writers is that great invention is about resolving the contradictory trade-offs in a system by a new technical arrangement, rather than simply managing them as an engineer [65]. Substance-field analysis offers some thoughts on the need to add compensators to a technology in its early phases, such as vibration dampers or pre-ignition aids to an engine. As a planning method, however, it carries a small weakness, namely the view of history as a pre-ordained process with uni-directional evolution. There is also more emphasis on the changing of widgets (single entities to multiple, touch interactions to field) than full-up architectures.

Beinhocker [40] and Mokyr [120] both understand the interplay between the base

of scientific knowledge, the skill of engineers, and the level of technology that currently exists. Further, Beinhocker describes a concrete the web of skills and competencies needed to participate in a community of technologists. Not only do the technology plans have to be developed, they have to be communicated in a form that practitioners understand. This implies not only that technologies are limited by the skills available to develop them, but also that the diversity of technical skills will also increase over time.

The biological representations of technology, including memetics, are useful in capturing incremental evolution, as this is about the improvement and refinement of existing concepts. Variation, selection and amplification can be seen to operate in multiple domains, including within a technological firm and in competitive markets. In some examples such as open software, even users are part of the community responsible for improving a given technology.

Conjecture #1 *Technological progress is gradual, and follows a pattern of natural mimicry and rearrangement of pre-existing technologies for new purposes.*

It is important to highlight a philosophical decision that is made at this point, and that it is made somewhat arbitrarily, hence the conjecture. The converse of this decision is essentially to throw up one's hands and rely on providence for solutions. The philosophical assumption that is undertaken in this thesis is that technology progress can be explained entirely by viewing it as a series of repurposings, recombinations and improvisations of other technologies, natural effects, or biology. This view is supported by a variety of authors [39], [40], [43], [68], [108], [133]. However, there are also a number of scholars focusing on technology that hold that serendipity and yet undiscovered mental processes produce technological creations that cannot be explained simply as changes to what came before [33], [56]. It should be noted that these scholars tend to focus more on the ecology of technology and strategic view of business, versus

the other set of papers that focus on specific examples of a technology's physical heritage.

The philosophical assumption that technology is essentially a process of remixing and adapting old inventions to new ends opens the door to predicting the future with the past. If old inventions are broken down to their most basic functions (for example, transistor crystals amplifying electromagnetic effects just as vacuum tubes do), then an analogy between multiple devices can be constructed. Once that analogy is created, then metrics of performance can be devised for that function. A candidate system architecture will then provide a set of functions and how they combine to meet a higher goal.

At this point, one may ask: if the past leads to the present, then why not simply utilize the data extrapolation methods discussed in chapter two? The critique of data based methods by Martino from the previous chapter still applies. In order for predictions from the past to remain useful, important conditions must not have changed over the duration of the prediction. For individual technologies without many input factors, this is likely not a major problem. For complex technologies with multiple performance parameters that can potentially be traded, a more nuanced view of a given system may be required. This is especially true in the case of design problems that lead to evolutionary acquisition approaches, since the relative importance of requirements can change through experience gained in field deployment.

A good example is given by Foster in his original book on S-curves [76]. In it, he discusses car tires, which have sub-components in their construction. In the section on tires, the discussion centers on different cord materials. Foster forms a composite performance parameter from cord strength, fatigue, adhesion and other properties (although he does not discuss weighting). But, he later explains that this is simply "a game within a game," as the cord technology competition happened concurrently with a competition between tire architectures, namely American bias-ply and European radials. Here, it is much more difficult to draw the S-curve, because once radials were developed

to have a sufficiently comfortable ride, their longer life became the selection criterion rather than smoothness of ride, which had been the driver for the development of bias-ply tires.

And this is where a leap is made. In the above example, the material properties of the cords are still the ultimate source of ride quality. They are simply drawn upon in a different fashion. Similarly, most of the mechanical properties of steam engines were retained in early electric motors, with the force generator (pulsing electromagnet) changing. Thus, a key to a new method will be to use data extrapolation for components, most of which will not change much, in analyses of new arrangements (architectures), which will change greatly.

Research Question #2 *How do the physical properties and effects of individual components combine to form technological systems?*

The above research question can actually be addressed by adapting standard systems engineering practice. The key is to start by thinking of functional decomposition. In a particularly good example, Dieter [62] gives a list of example functions for various mechanisms, such as nozzles, motors, gears, levers, etc. The function for each of these mechanisms is to increase, decrease, or transform one physical quantity into another.

The usefulness of physical quantities in this context cannot be understated. They are simultaneously universal and computable, since they follow known laws of physics or principles of operation for a given component. Since these quantities are based in the physical world, they are common across many implementations, and allow for the substitution of one for another.

To take the example of a train, the function of the engine would be to convert chemical energy into mechanical energy. The function of the transmission would be to divide the mechanical energy into the area of the wheels. The wheels serve to convert

rotational mechanical energy into traction upon the rails.

The train example adds another consideration. The system itself intends to transform something, namely its position from an initial to a final point. To do that, it uses its internal fuel store to produce a forward force outside of itself and propel it along the track. In order to do an analysis on how effective that function is, it is necessary to examine the interchanges between the various components (specifically the fuel tank and its effect on the train's weight) over time from start to finish.

Conjecture #2 *Functional decomposition techniques can be used to formulate any technological system as a set of physical quantity transformations between components.*

The general method for developing system components is now laid out. However, it remains to be seen how the full architecture should be specified in order to meet the goals of this research. The specification should enable easy arrangement of the components and be able to show a flow chart of various physical properties being melding into others until the system's output is developed. A further requirement for the specification is to be able to deal with incompatibilities between proposed components to be used within an architecture.

Also, as mentioned in the motivation section of this paper, it is vitally important to be able for the architectural model to be capable of handling multifunctional components. This is where the need for a new specification truly comes from. If the set of functions to be performed within an architecture remains constant, then the traditional tool of selection, namely the Morphological Matrix, is inadequate for the task. The Morphological Matrix relies on the ability to sort components into functional bins. And so, if the functional bins shift, split, and recombine, there is no longer a basis upon which to build the matrix.

Research Question #3 *How should the architecture be specified so that components can be easily interchanged and connected to each other so that their properties contribute to a system-level function?*

This question leads to search for inspiration from other places. The main inspiration comes from the work of Villeneuve [148], where a Morphological Matrix was reformulated into a graph in order to make it accessible to new combinatorial optimization techniques. The next step in the evolution of this formulation is to divorce the graph from the Morphological Matrix entirely, and impose new structures in order to handle the lost information from the compatibility matrix. The remaining question is in the form that these new structures should take.

The formulation to respond to this question is developed in much greater detail in the next chapter, including a formal mathematical structure. At this point in the paper, the response will simply be given below. It is also presented in the form of a conjecture, because it proposes a general method for expressing a given architecture that may not be compatible with all architectures it encounters. For now, it will be speculated to be the case.

Conjecture #3

The following structure can be used to formulate all system architectures:

1. A set of transformations that describe the transformation from one physical quantity to another by use of scientific principles
2. Connections that are specific to physical quantity to be exchanged between components
3. A pair of specialized nodes that emulate the use of nearby resources, the desired output from the system, and expulsion of wastes.

This conjecture is best expressed formally in the mathematics of graph theory, and is done so with explanation in the next chapter. The structure described above might be represented in other ways, but the graph theory provides a useful language and rigor. With the base model for the architecture devised, the next problem to tackle is how to sort amongst all possible architectures to find the optimum for an arbitrary fitness function.

Research Question #4

How does one find the optimum architecture amongst available alternatives with components at a given level of technological progress?

The first step in answering this research question is to consider the formulation of the representation of architecture given above. The architecture is represented as a graph, which makes it a well-defined collection of connections and vertices. This renders the problem of selection into a combinatorial problem, for which a few methods are available to choose from. The rationale behind the choice is covered in depth in the next chapter, which discusses the formulation of tools used in this research. The selection of the type of solution, in addition to considerations to adapt it to this specific problem, form the first

hypothesis to guide the research.

Hypothesis #1 *A genetic algorithm with mutation and crossover operators formulated in graph theory can find an optimum in the search space of possible architectures by evaluating a small portion of them.*

This hypothesis forms the basis for methods that are developed within this thesis. It is important to point out that this optimization problem is essentially an evaluation of alternative architectures, in either present or future time. The graph mentioned in the hypothesis represents an architecture of types of components and their interactions to achieve the functionality of the full system.

Each generation of technology, even down to the single compressor blade, brings a different trade-off between different dimensions of performance. A technology frontier [102] is a way of measuring this trade-off, in that improvement in one dimension of interest, e.g., work-enthalpy efficiency, can only be achieved by sacrificing another dimension, e.g., weight-work efficiency. Improving technology often breaks through these trade-offs and establishes new frontiers at higher levels of performance.

This framework is meant to integrate the historical information available for subsystems and components to develop the design space for the larger system. If the technology frontiers of any of these components improve over time, this framework will integrate those improvements to the system level. In this way, a system-level frontier can be constructed and its progress through time represented. In order to achieve this, the multi-dimensional growth curves formulated by Danner [57] will be employed.

By changing some characteristics of the components to represent technological advance, the door becomes open to evaluating the points at which architectures show similar performance. When the performance is very similar, this is the point at which a choice can be made to move from the incumbent architecture to an ascendant competitor.

Falsifiability Conditions

There are many ways for genetic algorithms to be ineffective in combinatorial problems. One way is for them to get stuck in sub-optimal parts of the search space. Another is to declare convergence too early. When getting stuck in certain areas, the algorithm may attempt to develop a large number of mutations or find crossover ineffective, meaning that many function calls are wasted.

The test for genetic algorithms in this case is to develop a number of example search spaces. It will be important to run the algorithm on multiple test spaces to see what traits its behavior displays. The number of possible combinations in the space can be known, as well as the relative optimality of all of the solutions. The genetic algorithm can then be executed and its run characteristics compared with those of the problem.

The next research question concerns the quality of solutions that can be developed through the genetic algorithm proposed above. The greatest question is if, in what is essentially a directed random process, there are enough useful results for the effort to be worthwhile. After all, it is not useful to have a program that generates an automobile where air from the tires is supposed to fuel the engine, or the engine torques away madly on the frame with nowhere for its horsepower to go. Thus, the following research question:

Research Question #4a *Is there a good way to ensure that a given architecture is "real," namely that the components are connected in a way that makes physical sense?*

The question is answered relatively simply, by considering the more structured Morphological Matrix problem. Even though the matrix contains every possible combination of the components within it, not every one of these combinations is physically possible. In order to account for this, a secondary matrix is used to store

information on which pairs of components are compatible and which are not. This matrix, formulated some years ago [100], is known as the Compatibility Matrix.

For the graph structure, it is possible to use physical properties as a means for screening which connections are sensible and which are impossible. For the automobile example, it only makes sense to connect the engine to the wheels via a drive train, with the common element being the torque provided by the engine. The wheels would then transform this into forward traction, which is carried through the frame to its payload hard points (the seats). This is embodied in the next design principle of the algorithm.

Conjecture #4

Realistic architectures can be maintained by labeling the inputs and outputs of components with the appropriate physical quantities, and requiring connections to be with input-output pairs of proper type.

This conjecture is the occasion upon which to split the research question it answers into two parts. The first part is a question purely of representation, to ensure that the physical architecture is matched properly to the input to the algorithm. The second part of the question is more mechanical in nature and concerns the algorithm's operation.

The second part of this research question is of proper representation of a physical architecture by a randomly (in the sense that architectures will be generated by genetic algorithm operators) selected collection of connections and components. If the representation is not complete, then the optimizer may chose a faulty representation with a strong score over a more realistic one with a slightly lower one. Optimizers are notorious for exploiting all of the rules of their creators to generate unexpected results.

An example to consider would be the layout for a bicycle, to be evaluated on minimal weight. The bicycle would have several components: gears to exchange pedal rotation speed for greater torque, a steering handle to change the direction of traction applied to the road, a chain to transfer pedal power to the wheels, a seat for the rider, and

a structure to hold the entire bicycle together. If, in the creation of alternatives, the algorithm does not connect the seat load to the bike structure, the structure's analysis would likely report a much lower weight due to lack of its principal load. In that case, the impossible bike would trump the real alternatives in the optimizer. This leads to the next research question.

Research Question #4b *Is there a good way to ensure that a given architecture is "real," namely that all inputs and outputs for the components interconnect?*

There are several ways to enforce interconnection between different parts of the graph. The goal in formulating an algorithm to enforce connections is to minimize interference with the genetic algorithm and avoid inducing bias in the results. Further, the goal of this work is to show potential processions from current architectures into potential successors. Thus, there is an assumption that the beginning point will be well-connected.

The specialized nodes, the source and the sink, also provide an avenue to assure that an architecture is viable. In many cases, the environment may provide a place to draw certain physical quantities to, or to send outputs. This would help ensure that at some level, each component has a properly connected input and output. A specialized operator, called rectification, was developed that searches for unconnected inputs and outputs within the architecture, and ties them directly to the environment. After crossover and mutation are performed, there is a second specialized operator, the symmetric partner to rectification called anti-rectification, that searches for input-output pairs of appropriate type that are empty. The operator then moves environmental connections to these new places.

Another way to enforce the reality of represented architectures is simply to set to zero outputs from components that do not have inputs connected (via a default setting

such as -1 on the input that would be overridden if it was connected to another component). In this case, the selection operator would then be responsible for screening out bad architectures.

Hypothesis #2

Using the selection operator to screen out nonsensical designs will be effective and cause minimal performance degradation, provided the program begins with well-connected parent solutions.

The speculation undertaken in this hypothesis is that the number of nonsensical architectures that results from the genetic algorithm's operation will be minimal. If it turns out to be untrue, then the structure of the genetic algorithm will have to be rethought somewhat in order to adapt it more specifically to this application.

It is important to consider that the problem at hand is in constructing architectures when the functional roles of various components are not well-set. If this was not the case, then a simple Morphological Matrix could be used to generate alternatives for evaluation and selection under various technological conditions. Thus, the algorithm is designed to take as naïve a stance as possible when connecting components, taking into mind only physical compatibility of the links.

Falsifiability Conditions

The rectification operator was originally developed in order to ensure that all inputs and outputs will connected, by reaching out to the source and the sink nodes. In the basic test design spaces, which will be presented in later chapters, the source and sink nodes contain all possible connection types. It was thus possible for all unfilled connections to be routed to these nodes in order to complete the architecture, and minimize the number of zeroed out solutions that were to be selected out.

The anti-rectification operator was developed from a desire for each operator to have a symmetric partner. The rectification operator biased the algorithm toward sending

connections to the source and the sink, and so the anti-rectification operator corrected this bias by moving connections back to the other nodes if possible.

For the test space that was used in this work, the combination of rectification and anti-rectification formed a type of completion operator. Their combined behaviors served to connect empty input-output pairs, which is a contradiction to the naïve approach described in Hypothesis Two. Thus, if the algorithm performs much better with these operators in effect than with them turned off, then the naïve approach does imply a performance hit relative to more structured alternatives.

The "begins with well-connected parent solutions" leads naturally to another hypothesis on the performance of the algorithm.

Hypothesis #3

If there is no initial architecture, the algorithm may start with a random selection of components. Further, if these components are assigned random (feasible) connections, it will greatly improve the performance of the algorithm.

Falsifiability Conditions

Challenging this hypothesis is straightforward. A test solution space can be developed as it is for the other hypotheses. With a known solution, the algorithm can be started with a number of components. In one case, the components would be connected randomly at the start, and in the other left unconnected. Performance of the algorithm in both cases can then be compared.

The next research question focuses on the components rather than the overall architecture.

Research Question #5 *What is the best way to forecast technological characteristics and trade-offs into the future?*

As discussed in the technological forecasting section of chapter two, there are a variety of methods to forecast technology characteristics in the future. Further explanation of the selection will be given in the next chapter, but the extrapolation methods class was the type chosen for use in this work. That left three major forms to choose from: hyperplanes, the Danner S-curves, and Inman's TFDEA. Since this is a central part of the method, testing is required to understand which would be the best choice for representing the effect of time on various technologies.

Hypothesis #4 *Of the three methods surveyed here (hyperplanes, multidimensional S-curves, TFDEA), TFDEA will be seen to be the most accurate.*

Falsifiability Conditions

This hypothesis is best tested by comparing the three methods against each other for a variety of different technological histories. This comparison was done by Inman for his TFDEA method against hyperplanes. For this research, as seen in the next chapter, a richer set of cases will be utilized to compare these methods.

A final research question comes from the nature of the combinatorial space that is represented by this approach to the problem. It will have a great deal of bearing as to which mutation effects will be most important, as well as serve as a guideline in navigating the combinatorial space.

Research Question #6 *What is the rough size of the combinatorial space of options in typical systems composed of a series of components chained together?*

In order to answer this research question, it is necessary to draw upon experience with technological systems. The physical quantities that are transformed within the systems must be considered, which would describe how many different types of connections are possible. In the example of the jet engine, every major component (fan, compressor, combustor, turbine, nozzle) conditions the air flow, which means that each component must allow an "air" type of input and output. Although there are a couple of other connections (such as fuel flow and work) must be considered, this air term effectively renders the graph a generic directed graph.

On the other hand, there are a variety of larger systems, such as aircraft, that allocate components to very specific roles. In this case, each physical quantity is directly transformed from one to another in a chain. In the aircraft, the wing transforms some of the momentum of the free stream into lift. The engine takes in onboard fuel and transforms its chemical energy into thrust by accelerating the free stream. The aircraft's structure transmit both of these forces to the payload and passengers to keep them moving with the aircraft. Each component then has only one role due to having only one set of feasible connections, and in this case each graph only contributes one combination. If this is true for all possible combinations of vertices, this reduces the combinatorial problem to the size of simply selecting components from a Morphological Matrix. This leads to a final conjecture. It can be supported or refuted by surveying a number of technological systems, but that will not be done in this dissertation.

Conjecture #5

The combinatorial space is bounded on the low side by the number of combinations of components, and is closer in size to that number than to the number of combinations due to a graph with an equal number of unlabeled vertices.

Summary

This chapter provides the centerpiece of this dissertation. It was written to concentrate the observations made of the problem and the literature review into a coherent philosophical posture by developing research questions. Many of these questions were answered by considering the literature review and providing conjectures. These conjectures may or may not be true, but they are difficult to test, and so must be incorporated into the philosophical posture of this work with hopes that some of the experimental work will vindicate them. The remaining research questions were developed into hypotheses that make educated guesses about the performance of various aspects of the algorithm that will be laid out in the next chapter. These hypotheses will require the exercise of test cases and experiments to be supported.

The philosophical posture of this work is one of evolutionary gradualism, namely that every technological system has an antecedent either in the natural or the artificial world. This is not to say that technological improvement is trivial – quite the contrary, this posture is a declaration that innovation is so difficult that it can only be achieved by improvising changes to gains that have already been made. This particular posture also utilizes one of Darwin's thematic insights [60]. He knew that evolution would be a very difficult theory to conceive or defend, and so he aimed to explain evolution *in medias res,* rather than its ultimate genesis.

The four hypotheses put forth in this thesis claim that it is possible to use an optimization heuristic to provide insights into this process, and to probe the points at which an incumbent architecture may be supplanted by a new configuration. Further,

they make statements about the factors affected its performance, all of which are testable by employing it on sample problems and taking a measure of the results.

CHAPTER 4

THEORY AND FORMULATION

This chapter will describe the basic set of models and tools that have been drawn upon in order to develop an algorithm and a method to find the points of transition between incumbent and ascendant architectures. The theoretical basis for each element of the algorithm will be laid out, along with choices that were made as it was formulated. In addition, studies of supporting tools will also be described.

The method described here revolves around four basic tools that are to be used together in order to forecast potential times for the relative prevalence of different architectures to shift. The first major piece of the method is the structure in which architectures can be expressed consistently. The second piece is to develop forecasting models for components that are combined to form architecture. The third piece is an automated search algorithm that can find optimal architectures based on the performance levels of their constituents. Finally, the results of the algorithm must be visualized as a response to inputs to the component forecasting. This also opens the door to visualizing variants and "what ifs" from the data loaded into the search algorithm.

Although the method is segmented into four tools, one part has received the great majority of effort in both formulation and implementation. This would be the selection algorithm, which is part of a software code named Sindri, who in Norse mythology were responsible for the fabrication of three of that mythology's most important artifacts: Freyr's magic boar that could light the dark and run faster than any horse; Odin's magic ring that produced infinite wealth; and Thor's legendary hammer Mjolnir. Sindri was aided in his efforts by his brother, Brokk, which is the name given to the larger method. The names seem fitting for an algorithm and a method that deal with advances in technical skill.

The method will be presented element by element, with the theoretical considerations discussed first, and the implementation of these ideas immediately following. The material is presented this way in order to provide the reader both a theoretical rationale for various design choices, and to immediately show the consequences in the resultant program. This risks obscuring how the method as a whole was conceived, and so this global view will be presented up front.

Genesis of the Method

The general sweep of the method was sketched out by considering again the base problems. There are two basic use cases that have been described in this dissertation. The first is the evaluation of a large number of architectures when there is no clear incumbent. The second use case is to understand the conditions under which an incumbent architecture is vulnerable to an upstart. Since there are a large number of potential upstarts at a given time, this use case has similar requirements to the first, except that the sensitivity to assumptions should be given greater weight in making strategic choices.

The base problem then is the ability to compose and compare multiple architectures on the basis of quantitative merits (performance, cost, etc.) under a variety of technological considerations and assumptions on the rate of progress. The composition and evaluation of architectures should be automated. The ability to generate these architectures must be flexible enough to account for multifunctionality. As discussed in the previous section, this requirement necessitates the abandonment of approaches that rely upon Morphological Matrices. And finally, the end product of the method must not be a simple point solution, but rather a family of outcomes that informs the user of the contours and trends of a given situation.

Software Framework for Implementation of Architecture Search

In order to implement the automated search for a desired architecture, it was necessary to choose a base software to build from. Ideally, the software would have all necessary features and simply require re-interpretation or simple data input in order to provide the implementation. This is not the case, and so the selection criteria for the software were to maximize the pre-existing support for these concepts, to have the easiest means for extension of this support into the final implementation, and to have good execution characteristics.

There were three major options that were considered for use: ModelCenter, iSight, and Pacelab Suite. All three are designed as integration environments, although each has its own strengths and weaknesses in terms of functionality. The most important aspect in this case was extensibility, which is held by Pacelab due to its Application Programming Interface (API). The API offers control of almost all software functions, including the ability to add and connect analyses, which is necessary to construct representation of architectures on the fly from component analyses.

In addition to providing the ability to compose architectures on the fly from modular analyses, the API affords the ability to implement custom data structures necessary to facilitate proper connections. Each component is labeled with the number and type of connections, which provides cues to the optimization algorithm as to which combinations of components and links are proper and which are nonsense. Another use for custom data structures is to form “vectors” of quantities that come with the transmission of working fluids (thermodynamic properties, mass flow) or external fields throughout a given system during its use. The vectors help to retain the physical connection between these quantities and also ensure that the structures of different connection types are interpreted equally by the optimization algorithm.

Details of the software implementation will be provided in the next sections of this chapter under the appropriate headings.

Graph Theoretic Description of Architecture

As discussed at the beginning of this paper, architecture is defined in terms of base components and their connections and interfaces. The connections and interfaces naturally correspond to vertices and edges of a graph structure. While this model is adequate for simple drawings and networks, it is too poor to describe a physical architecture. A physical architecture needs to embody considerations of interfaces, physical transformations and physical quantities, along with some parameters that describe the design variables available for adjustment by the designer. A search for inspiration in enriching this graph can be found by consideration of a few other cases where graphs were used to specify technological architectures.

The basic graph is defined by two sets, the set of vertices and the set of their connectors, which are further defined as unordered pairs of the vertices they connect. Formally, this is:

$$G = (V, E), \quad V = \{v_1, v_2, v_3 \dots v_n\}, \quad E = \left\{e_1, e_2, \dots e_n : e_i = \left(v_j, v_k\right) | v_j, v_k \in V\right\}$$

One of the original inspirations for this work, the transformation of a Morphological Matrix into a graph by Villeneuve [148], does not enrich the graph per se, but deals with it in a raw state. However, the graph serves as a selector for technology options, which are represented by nominal values (e.g., thrust/weight, Isp of rocket engines) and sent to an integrated launch vehicle analysis [149]. The analysis engine, RASAC, constructs the launch vehicle design problem from cues in the graph. This analysis also contains a set of design variables that are modified to find optimal (minimum weight) vehicles subject to the technological factors implied by selections on the graph version of the Morphological Matrix. By considering the analysis, the technology factors attached to each alternative node on the graph could be considered an enrichment of the graph structure. Formally, this might be expressed with a new set of variables that is mapped to each vertex:

$$\mathbb{K}: V \mapsto K$$

As an example for this notation, $\mathbb{K}(v_1) \coloneqq \{k_1, k_2\}$ where k_1 is Isp and k_2 is thrust/weight ratio on a vertex that represents a rocket engine.

Another example is due to Blickle et al [44], which is a specification designed to provide for automated design of computing architectures. They state an interest in taking computer-aided techniques that had been successful in VLSI design up to the level of computing architectures. They define the design problem as allocation (selecting one from multiple possible graphs), binding (fixing connections between components to represent operations) and scheduling (accounting for communications delays between nodes). The enrichment of the graph with sets for allocation, binding, and schedule is uninteresting, as it merely applies integers to edges and vertices to represent inclusion. The interesting enrichment is that of pin count, shown as a function below. The pin count of various chips is used in the optimization algorithm to determine whether a potential connection was feasible or infeasible.

$$pin: V \mapsto \mathbb{Z}^+$$

A more formalized structure for physical architectures is the Formal Design Theory, presented by Braha and Maimon in their text [48]. They define an "artifact space," which is meant to represent all of the technological objects currently under consideration in a given design process. This artifact space is formally represented as:

$$\mathcal{D} = \langle M_0, C^0, M^* \rangle$$

where $\mathcal{D}$ is the design space, M_0 is the set of atomic modules, C^0 is the set of connections, and M* is the set of modules made up of composites of atomic modules and their connections.

The Formal Design Theory also defines a function set that steps between the artifact set and performance characteristics of interest:

$$\Gamma: \mathcal{D} \to \mathcal{F}$$

The functional set C is also described by Braha as a "transformer," which

represents the conversion of effort from one medium to another or flow from one medium to another. This provides another level of validation for the idea that systems can be represented by a series of physical conversions.

The strongest inspiration for a graph representation in this work comes from Emmerich et al [69], which describes an optimization of a chemical production network. The notional plant they work to optimize is a series of condensers, distillation columns, furnaces, and so on. The goal of the optimizer is to provide the proper number and connections between these components. And the optimizer operates upon a structure the authors call a structure graph, with the following definition:

> **DEFINITION 1:** *An S-Graph is a tuple* $(V, V_f, E, n_{in}, n_{out}, u^o, O)$*, where* V *is a set of vertices,* $V_f \subset V$ *is the subset of fixed vertices of* V, n_{in}, *and* n_{out} *are functions of the type* $V \mapsto \mathbb{N}_0$, E *is a set of edges with* $E \subseteq \left\{((v_1, i), (j, v_2)) \mid v_1, v_2 \in V \wedge i \in \{1, \dots, n_{out}(v_1)\}, j \in \{1, \dots, n_{in}(v_2)\}\right\}$, *and* $u^o: V \mapsto O$ *is a function that assigns an element of finite set* O *of operation-types to each vertex.*

The above definition has nearly everything that is required. It has a function map to transform vertices into analyses (which are exercised in the article referenced), an encoding to account for the number of inputs and outputs, and fixed vertices to represent sources and sinks. The only thing that is missing is the ability to label the edges, inputs, and outputs with appropriate physical quantities, which is a straightforward addition to make. This, along with a vector of design variables, leads to a graph definition that will be used for the rest of this work.

> **DEFINITION 2***: An S'-Graph is a tuple* $(V, v_{source}, v_{sink}, E, p_{in}, p_{out}, u^o, O, P, X, Y, K)$*, where:*
>
> V *is a set of vertices,*
>
> v_{source}, v_{sink} *represent connections into and out of the bounding box*

around the system, respectively,

p_{in}, p_{out} *are functions of the type* $V \mapsto P^n$, *where* n *is the number of allowable connections in or out of the vertex,*

E is a set of directed edges with $E \subseteq \{((v_1, i, p_m), (j, v_2, p_m)) | v_1, v_2 \in V \wedge i \leq poutv1, j \leq pinv2 \wedge pm \in poutv1 \cap pinv2 \in P$,

$u^o: V \mapsto O$ *is a function that assigns an element of finite set O of operation-types to each vertex, and*

P *is the set of possible types of physical interactions among vertices,*

X *is a vector of values* $X = \{X_1, X_2, .. X_n\} | X_i \subseteq \mathbb{R}^{l_i}$, *where* l_i *is the number of inputs to vertex i in the graph,*

K *is a vector of values* $K = \{K_1, K_2, .. K_n\} | K_i \subseteq \mathbb{R}^{m_i}$, *where* m_i *is the number of technology factors on a vertex i, and*

Y *is a vector of results of* $Y = \{Y_1, Y_2, ... Y_n\} | Y_i = O_i(X_i, K_i), Y_i \subseteq \mathbb{R}^{r_i}, O_i = u^o(v_i)$, *where* r_i *is the number of outputs from a vertex i.*

An illustration of a notional graph following this structure is provided in Figure 12.

There are a series of layers added to the graph above, but it is important to consider that this is simply the formalization of what many integrated design environments, such as ModelCenter, Pacelab, or iSight, usually present to users when linking together analysis modules. The discussion of the implementation of the graph in software that follows will hopefully illuminate the above definition more fully.

Software Implementation of Graph Structure

Each element of the tuple $(V, v_{source}, v_{sink}, E, p_{in}, p_{out}, u^o, O, P, X, Y, K)$ described in the definition of the S'-Graph is mapped to a different aspect of the Sindri software. Some of this is done with native aspects of the software, the rest is due to work in the API.

The first element of the tuple to address is the set of vertices, V. The vertices are meant to represent components, with specific variables and physical actions associated with them. The appropriate element for this in Pacelab Suite is the Engineering Object. The Engineering Object is a basic object that serves as a container for a variety of user inputs. It enables the user to specify a name for the object, parameters of a variety of types, definitions for geometry, executable code in the C# language, and macros to relate parameters on the basis of either C# or simple formulas. Once an Engineering Object is defined in the Knowledge Designer part of the Suite, it can be instantiated any number of

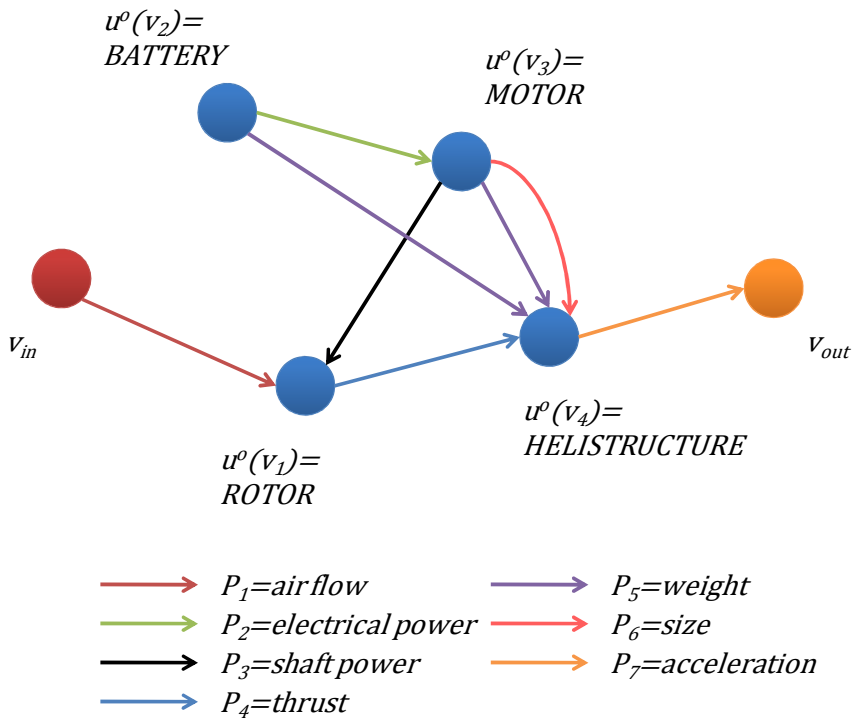


Figure 12. Example of S'-Graph

times within Engineering Workbench, the integration part of the Suite.

The executable code and algorithms written in C# within the Engineering Object forms the basis for expressing the u^o function. The structure of the Engineering Object means that operation types (the algorithms that model the transformation of one set of physical quantities into another by a real component or subsystem) can be specified very easily by the user. The way that the user chooses to connect the different parameters together with these macros also defines the assignment of Engineering Object parameters to either the X, Y or K sets. The main difference between the X and K sets are that the elements of K are meant to be affected by the progress of a time parameter attached to each of these objects.

v_{source} and v_{sink} are specified through the development of Engineering Objects given the names of "Source" and "Sink." These entities are searched for within the Sindri software every time there is a change to be performed on the graph and treated specially. They are the only vertices that can have the numbers of their inputs and outputs changed and assigned to different physical quantities (albeit from a pre-specified list) at runtime.

The edges of the graph are to be provided by Functional Objects, which are the other basic prototype available to Pacelab users. The essential difference between the EO and the FO is that the FO does not contain any "native" parameters that can be assigned values. The parameters of the FO are simply there for attachment to EOs – as the name implies, these objects are meant to house functions. It is a convention of Engineering Workbench that only FO's or formulas can connect the parameters of different objects, and so a generic FO will be used to serve as a universal edge within the Sindri program.

The final entities of the tuple, p_{in} and p_{out}, do not have a native formulation within Pacelab. The API was thus used to develop a custom data structure to house the specification of inputs, outputs, and their physical types for the main algorithm. This data type was termed the "Tag container" and was developed as a transparent plug-in to Knowledge Designer. The screenshot of Figure 13 shows an example dialog box for how the user specifies the inputs and outputs of a given EO. The specification is a simple exercise in applying textual "tags" to the inputs and outputs, which are then interpreted by Sindri in order to restrict which vertices can connect to others based on this labeling.

In addition to the specification above, there are some practical rules and limitations imposed upon the graph. The main rule is that the graph is currently limited

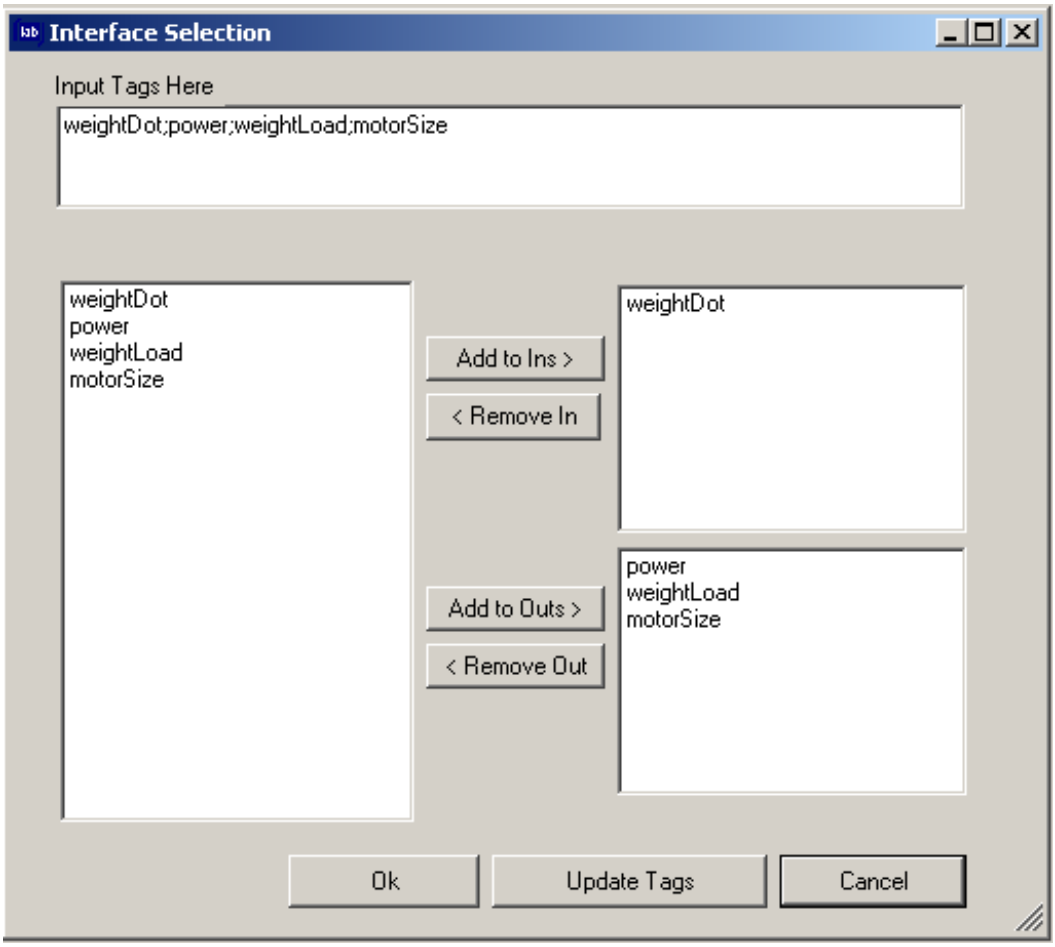


Figure 13. Tagging dialog within Sindri

to being acyclic, as it makes evaluation much more straightforward. This is not a paralyzing limitation, but it does restrict the types of architectures that can be modeled. A closed-loop power system would not be well-defined, for example.

Architecture Search Problem Formulation

With the graph structure specified, it is possible to formally specify the problem of finding the best architecture from its alternatives under a series of conditions. This specification will lead naturally to a hunt for the correct tools to address the problem.

To begin with, it is worth restating the selection problem is more precise terms than it has been so far. Using the language of the graph structure developed in this chapter, the problem of selection is as follows:

Given a set of vertices $\tilde{V}$, a series of functional assignments u^o to functions O, a series of connection types P, and a set of values for the vector K,

Find an S'-Graph structure $(V, v_{source}, v_{sink}, E, p_{in}, p_{out}, u^o, O, P) | V \subseteq \tilde{V}$ *and settings for the vector $\hat{X} \in X$, where $\hat{X}$ is the set of input variables available to the designer, such that $Z \coloneqq f(Y)$ is maximized.*

In the problem above, it is important to remember that the vectors Y, X, and K have their sizes determined by the vertices that are currently in play. This implies that it may be possible to separate the problem into two pieces: a discrete optimization problem to find the proper subset of vertices V from $\tilde{V}$ and the edges E to connect them, and a continuous optimization problem to find the appropriate settings for $\hat{X}$ once the graph structure has been specified.

As mentioned in the previous chapter, it was hypothesized that a genetic algorithm that could operate on graphs would be able to efficiently sort through architectural alternatives and find the one with the best objective. Both the implementation of this particular algorithm and the thought process behind its selection will be presented in this section.

The above problem takes the specification of K, O, and P for granted, which will also impact the value of the objective function Z. Thus, in order to have a fully specified problem, it will be necessary to generate a basis for these mathematical objects. These objects were chosen to represent technology, component modeling, and the types of physical exchanges, respectively. In order to have a complete method, these objects will have to be considered. This will be explored further in this section.

Technique Downselection

Turning attention toward the architectural problem indicates the next step, which is to properly classify it. Since it involves the specification of a finite set of discrete variables, it is a combinatorial problem. In addition to being formulated as a graph, the problem can be thought of as an integer programming problem. The reason for this is that each vertex type could be assigned an integer index for type, an integer count (in case more than one copy of vertex can be used), and the connection of a potential edge could be signified by a 0 or a 1. This classification is very important, because of a computational property associated with integer programming. It has been proven to be NP-hard [126]. In other words, it is as least as hard to solve as a nondeterministic polynomial (NP) time problem. NP combinatorial problems have the (likely, but unproven) property that the only guaranteed solution is an exhaustive search of all possible solutions.

In the face of this, there are nonetheless multiple methods that can attack this problem in far fewer trials in most cases (there is still no mathematical guarantee of a solution, but they perform well enough to be used often). These methods are the Branch and Bound, evolutionary algorithms, ant colony optimization, and simulated annealing. These techniques are typically called heuristics, since they do not have a rigorous mathematical basis for solving problems, but have been shown to be effective with high probability.

Branch and bound techniques [107] are fundamentally about splitting the design space into either/or regions to consider. A bound for the optimal value is then developed and nodes are expanded that have results within this bound. It can be used on both graph and integer problems, with the prior implemented by considering graph solutions with or without certain edges being used. However, it is not clear as to how the ability to work with constraints on connections would be properly handled.

Simulated annealing is a technique that attempts to get explore a design space broadly by simulating the activity of atoms in hot metal. The algorithm starts with a number of randomly generated solutions. At the beginning, "temperature" is high, which means that large changes are made to the starting solutions. After these changes, the candidate solutions are evaluated, and candidate solutions are kept if they show improvements in their objective functions. As the "temperature" goes down, the moves become smaller and the algorithm eventually ceases. However, it is worth noting that there is no way to combine features of good solutions, which can make it difficult for the method to efficiently cover large spaces. Also, in considering that new architectures are often developed by hybridizing successful alternatives, this combination would seem to be a desirous trait in an algorithm.

In formulating this method, ant colony optimization [67] was given a great deal of consideration. The fundamental design of ant colony optimization is to emulate the pattern of ants pursuing food, and selecting from a variety of paths to do so. Along the way, a pheromone is laid down. Since the shortest paths have the ants finding their food most rapidly, they will tend to collect the most pheromone, and self-reinforce until the ants travel along one or a small number of paths from nest to food. This metaphor is applied to computation by substituting the fitness function of a given combination for the distance to the food.

Ant colony optimization's primary mode of operation is to start with a graph that contains all possible routes (such as city pairs in the Traveling Salesman Problem) and

select from it an optimal subgraph. This subgraph selection is something to examine in order to understand the suitability of ant colony optimization for this problem. The important question is how to develop the graph of all possible connections upon which to place the ants. This is not as trivial as it may seem, as it requires a consideration of both the types of edges and the allowable subgraphs that would result from an ant traversal.

The first consideration is that the ant solution must be either a path (with start and end conditions) or a tour on the graph. A tour is a sequence of edges and vertices that returns to its start point, while a path starts and ends at two distinct vertices. This is in keeping with the ant analogy. It would also seem to be a first mark against the use of this method for this problem. However, this restriction can be relaxed if it is realized that an arbitrary graph can be composed of a union of paths and tours. The difficulty in that point of view is to decide on where to restart the ants.

The second consideration is whether there is a guarantee that subgraphs derived by this method will be feasible solutions to the problem. There is not, as can be illustrated in the following counterexample, which is portrayed in Figure 14. The problem involves a sink, a source, and four other vertices. There is only one edge type to consider. A variety of possible graphs (G_1*, G_2*, G_3*) are combined into a single supergraph for the ant colony optimization to operate upon, GI. However, at least one of the potential paths for the ants does not include all of the vertices. Since the vertices are selected in order to be linked, this is not an acceptable result. Also, since the decisions that make up the ant colony procedure are based on local information (e.g., to branch one way or another from a given vertex), there is not a good way to enforce rules for passing through all vertices.

With all of the other classes of techniques eliminated, the final body to consider is the evolutionary algorithm. There are many different types of evolutionary algorithm, including the genetic algorithm, evolutionary program, genetic program, and the evolutionary strategy. The two programming types are traditionally used to develop

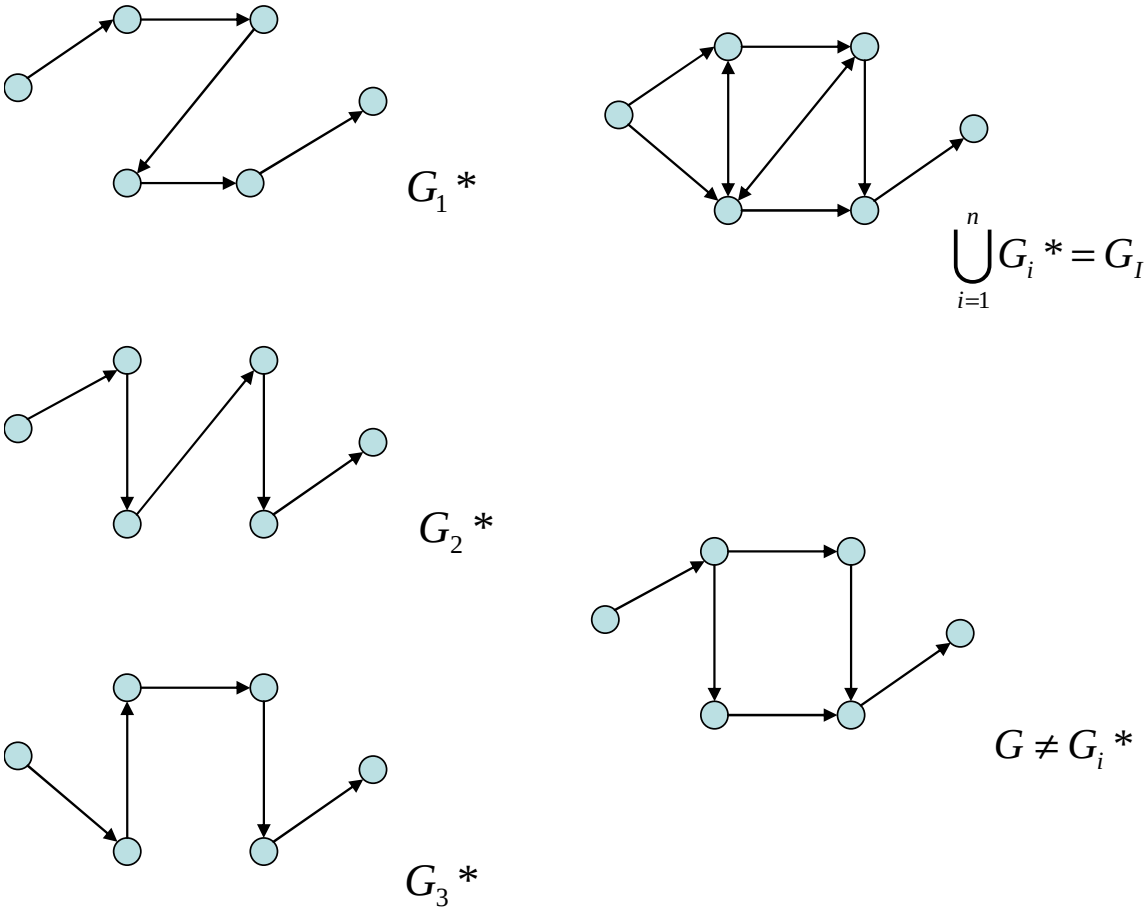


Figure 14. Attempting to refind eligible graphs as subgraphs of union graph

computer programs through evolving a series of commands and basic operators (AND, OR, mathematic operators) to arrive at a desired program [106]. These techniques have also been used to develop a variety of circuits, including the reproduction of several that had won patents and awards [105]. This method has also been used to patent physical objects such as lenses [135].

The evolutionary strategy was developed in Germany [41], and is constructed in a way that is similar to the genetic algorithm. It has mutation, selection, and crossover operators like the GA. It tends more to deal in real values than the genetic algorithm, although it is compatible with finite values. Also, genetic algorithms have real value variants, which also smudges this difference. The major difference between the two methods is in the way reproduction is handled. In genetic algorithms, reproduction is

done based on fitness (either deterministically or probabilistically) over the entire population. In evolutionary strategies, there is a much stronger concept of parents and children. There is a parameter, λ, that is used to determine how many candidates are generated from each point in the population. These candidates are then mutated and subjected to crossover, and then compared to their parent based on fitness. The selection is then made among just the candidates within this group (again, either deterministically or probabilistically).

In this case, the algorithm of choice was what is called here a genetic algorithm. However, because of an interest in warding off premature convergence, the reproduction style of the evolutionary strategy was employed. The main reason to call this algorithm genetic is that there is still a chromosome encoding of a graph that is operated upon to develop new graphs.

The goals of developing this algorithm should be contrasted to other genetic algorithm variants that have been developed recently. One in particular is the gene-corrected genetic algorithm employed by Raczynski [128] for selecting options from a vector of potential technological enhancements. Since not all enhancements were compatible, it was necessary to eliminate members of the population that contained incompatible combinations. This was done by randomly choosing a member of the incompatible combination to leave in the selection while de-selecting the remainder. Rather than explicitly searching for graphs that represent nonsensical architectures, the preferred approach in this work is to let the connection logic of the graph automatically show a poor fitness, such as when a given component has no inputs and thus has an output of zero.

Another algorithm to compare the graph approach to is the structured genetic algorithm, such as was applied by Buannano [50] in order to search the space of possible high-speed aircraft configurations. In the structured approach, part of the genetic string activates or deactivates other members of the string. This mimics biology, in which there

are genes that turn regulate the expression of other genes. For the graph approach, however, the goal is to express the structure within the graph. Thus, unlike a structured approach which maintains a constant gene length, the work undertaken in this thesis allows a chromosome of varying length to be used. This is done by adding a "dimension" to the chromosome, namely letting every vertex have a string of varying length assigned to it. This is discussed further later in this chapter.

It is also important to consider that in this problem, there is a definite starting point, namely the current state-of-the-art architecture. Thus, it is worth considering the idea of constructing a graph that is a neighbor to the original, namely that many of the vertices and edges are similar to those of the starting point. These changes would represent the evolutionary successors to the starting point. This does not mean that more revolutionary steps cannot be modeled in this way, it simply means that the antecedents of those particular devices should become the starting points. Either way, this way of viewing the problem is what suggests the use of a genetic algorithm as the optimization technique of choice.

Operator Specification

The decision to use a genetic algorithm as the basis for searching the combinatorial space of architectures leads naturally to a new set of decisions that must be made. A genetic algorithm's operators, mutation, selection, reproduction, and crossover must be specified. In addition, and as alluded to earlier, a new operator, rectification, is employed in order to deal with the requirements that an architecture has its components connected either to each other or to the outside environment. Each of these operators must be specified in the context of the graph structure that was laid out earlier in this chapter.

There are a series of principles that have been developed over time to guide the development of operators for evolutionary algorithms in order to improve their

performance and keep them well-balanced [41], [69], [130], [131]. *Reachability* states that it should be possible to reach any other point in the design space with a finite number of mutations. Sensibly, this means that it is possible for any initial design to be transformed into the global optimum. In some cases, this is taken further to become *ergodicity*, meaning that any point in the design space can be, with finite probability, found from any other point by a single application of the mutation operator. *Symmetry / unbiasedness* is the property that every operator has a symmetric pair that can reverse the change that the operator has just applied, or that the same operator can reverse its own action when applied twice in a row. *Scalability* requires that the mutation operator can have its "strength" tuned in order to adapt to different search spaces and the relative merits of moving quickly or slowly across them.

There are also a number of principles that apply only to the crossover operator. *Assortment* requires that all combinations of features of the parent individuals should be available to the offspring. *Respect* and *transmission* stipulate that both parents contribute features to the offspring and that each feature of the offspring has an antecedent in at least one parent, respectively. Finally, there is *linkage*, that implies that grouped features should be inherited by offspring with a similar relationship to each other they had in the parent.

This seems like a long list of requirements for an algorithm to simultaneously satisfy, but most of them come fairly naturally. In the graph representation, rather than a string of values or binary bits, the structures of connection are already present. They can be interpreted and used in order to guide the development of a crossover operator. Manipulating the graph provides clues to develop a mutation operator.

Mutation

The mutation operator is a composite of multiple types of graph changes. Each of these changes can be assigned a probability, which is to add up to one. Thus, the operator

can be weighted toward one particular type of mutation over another. There is also a "mutagen" scaling factor, which influences how many elementary mutations are performed on a given graph in a given generation. This scaling factor is implemented as the variance of a normal distribution. Each time a graph is mutated, a random number is drawn from this distribution, rounded to an integer, and used as the number of mutations to perform.

The elementary mutations implemented in the Sindri software are:

- Adding a vertex to a graph, potentially by splitting an edge and putting itself in the "middle" of two vertices
- Removing a vertex from a graph and deleting the edges connected to it
- Adding an edge to a graph
- Removing a vertex from a graph
- Swapping the heads of two edges
- Swapping the type of a vertex

When these actions are performed, there is also a check for compatibility. An added vertex can only place itself in the middle of an edge (splitting the edge in two and connecting the new head and new tail to itself) if there is both an available input and output of the edge's current type. Similarly, edge head swaps are only legal between two edges of the same type. Finally, when the type of a vertex is changed, the new vertex must have inputs and outputs compatible with the edges that the original vertex had connected to it.

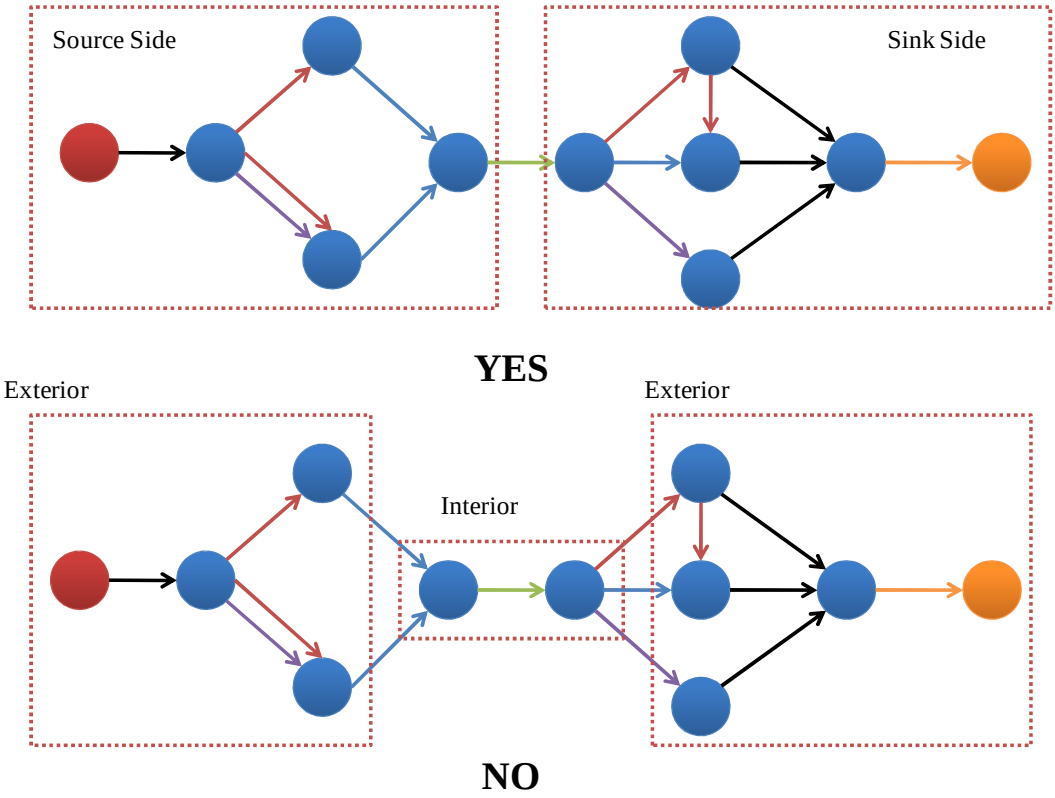


Figure 15. Comparison between bridge-based (legal) and interior/exterior crossover (illegal)

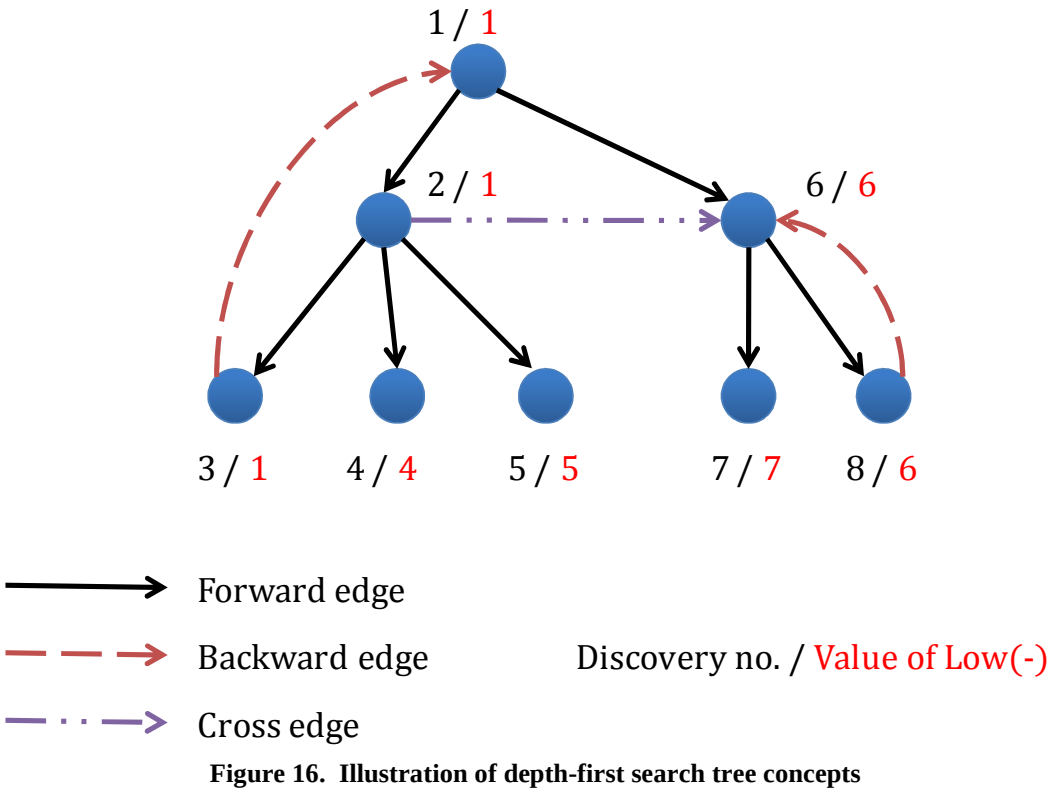


Figure 16. Illustration of depth-first search tree concepts

The reachability of this operator can be shown by considering that an arbitrary graph can be built up through successive additions of edges and vertices from a null graph. It can also be built through a succession of removals from a graph in which it is a subgraph or simply altered by vertex and edge swapping. The mutagen parameter satisfies scalability. And the fact that the number of mutations is developed from a normal distribution means that an arbitrarily large number of mutations could be performed within a single generation. Combined with the property of reachability, this yields ergodicity. Finally, symmetry is represented by add / remove pairs of mutations and is satisfied by two applications of the swap mutations.

Crossover

The crossover operator was designed to respect the graph's connectivity as much as possible, and to break the graph into two pieces for recombination. Unlike the typical bit-encoded crossover, there is no "interior" crossover possible. The interior versus exterior crossover is illustrated in Figure 15. For the graph-based operator, two graphs are split into two sides, the "sink side" and the "source side," and recombined to form two new graphs. This operator does not account for the ability to have an "inside" and "outside" segment, as illustrated in Figure 15. This was a choice made for implementation simplicity.

In order to find where to divide the graph, the concept of a graph bridge is employed. A bridge is any edge whose deletion would cause the graph to become disconnected, meaning that it is not possible for all vertices to be reached from another by traversing graph edges. In order to detect bridges, a standard algorithm [32] for finding articulation vertices (a vertex whose removal disconnects the graph) is used. The algorithm is executed by developing a depth-first search tree Figure 16 that starts with the source vertex as the root. Then the tree is traversed, and the order in which vertices are reached is recorded, which becomes the discovery number of the vertex. The operation

low is then applied to discovery numbers:

$$low(v) = min(disc(v), disc(w) : (u, w) \text{is a back edge from descendent } u \text{ of } v)$$

In the definition above, a back edge is any edge that connects a vertex back "up" the tree from a vertex of higher (later) discovery number to one of lower (earlier) discovery number. Thus, the low operation is meant to discover the lowest discovery number that is reachable from the descendents of vertex. If a vertex descendent is connected to a antecedent of that vertex, that means that there are at least two ways to reach that vertex in the original graph.

Any edge that has two articulation vertices at its ends is a candidate to be a bridge. The same information used to find the articulation vertices is used to check if this edge is truly a bridge.

For the purposes of the search for bridges, the S'-graph is "flattened" into an undirected graph, and if multiple edges connect a pair of vertices, they are rendered into a

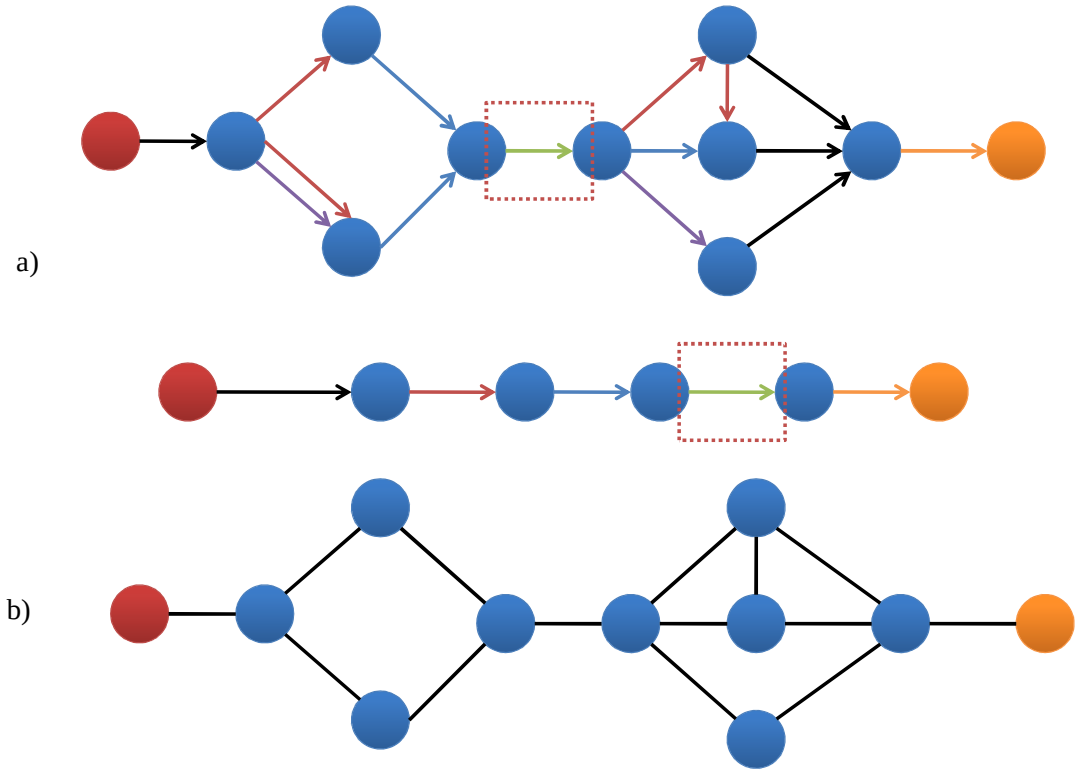


Figure 17. Crossover process in multiple steps, a) with two graphs entering crossover and b) top graph rendered as undirected graph. The boxes show the bridges that will be the meeting point of the crossover

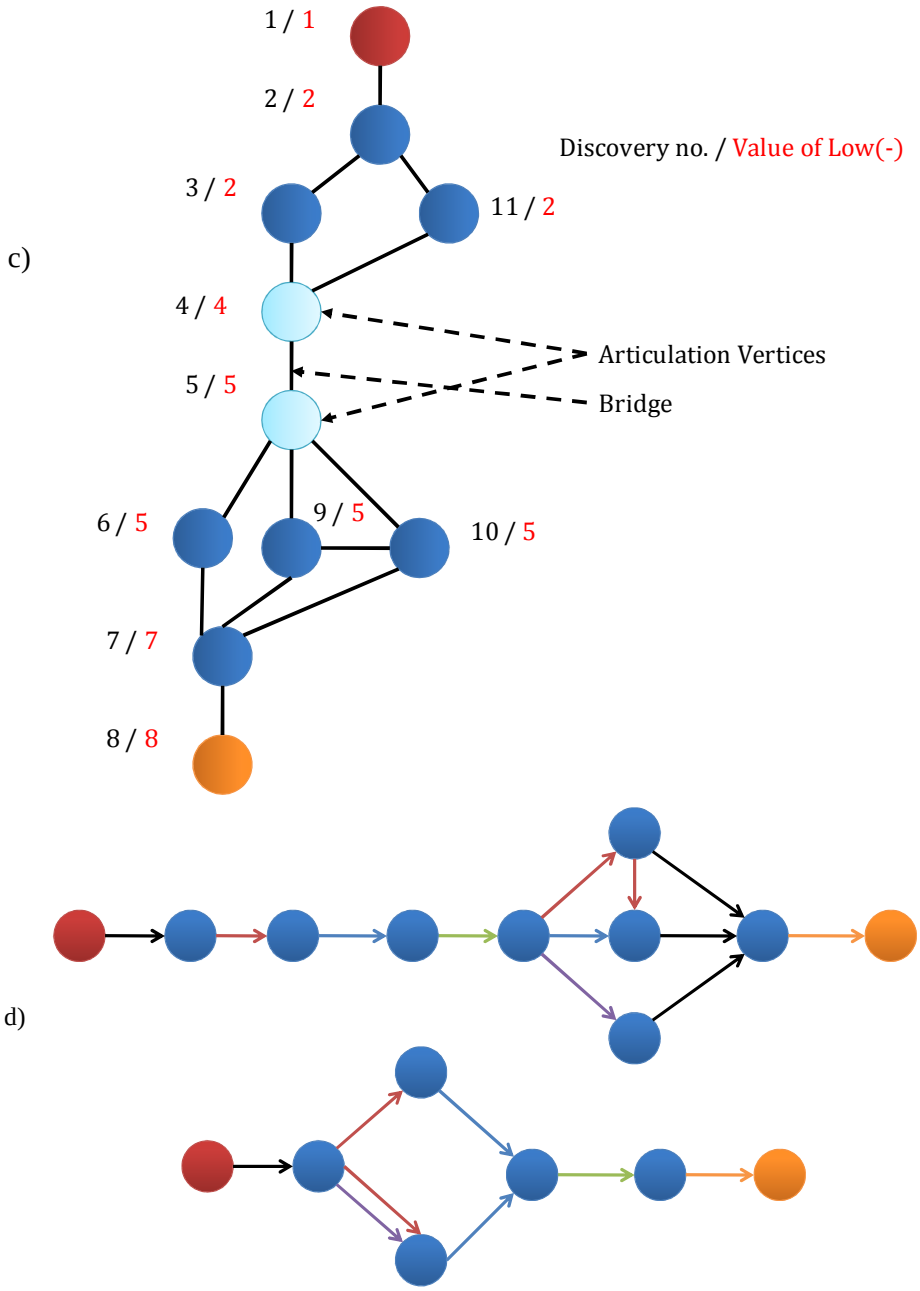


Figure 18. Depictions of c) depth-first search tree, and d) competed crossover results

single edge. Once all of the bridges in both graphs submitted to the crossover operator are found, one bridge from each is randomly chosen. The types of the edges that were flattened into a single, undirected edge, are taken into account as well as their direction. As illustrated in Figure 17, this information is used to ensure the compatibility of the new graphs that are generated by putting the fragments of the old together. The formation of the depth-first search tree and the results of the crossover are depicted in Figure 18.

The design of the crossover operator satisfies respect, transmission, and linkage, the properties that concern the relationship of graph features before and after crossover. Since the two graphs that result from crossover are composed wholly from the graphs that go into the operator, the total number and types of vertices are conserved, as are all of the edges due to the compatibility check. However, the fact that this operator is based upon finding bridges, and does not have a way to take "interior" fragments or deal with cycles in the undirected graph, means that the property of assortment is not satisfied. There is an indication that crossover on cycles is possible due to research on genetic algorithms for chemical engineering [81]. This will be considered an area to be improved through further research.

Selection

The selection operator is very straightforward. It is taken from evolutionary strategies, where there is a parameter, λ, that identifies the number of children a given member of the trial population will have. These children and the parent then compete to become the new (or retain the status of) parent in the next generation of execution. This is a deterministic operator, selecting the candidate with the highest fitness to become the new parent. This is illustrated in Figure 19, where three children are generated, varied, and then evaluated. Based on the fitness values of the parent and children I through III, child I is chosen by selection to become the new parent.

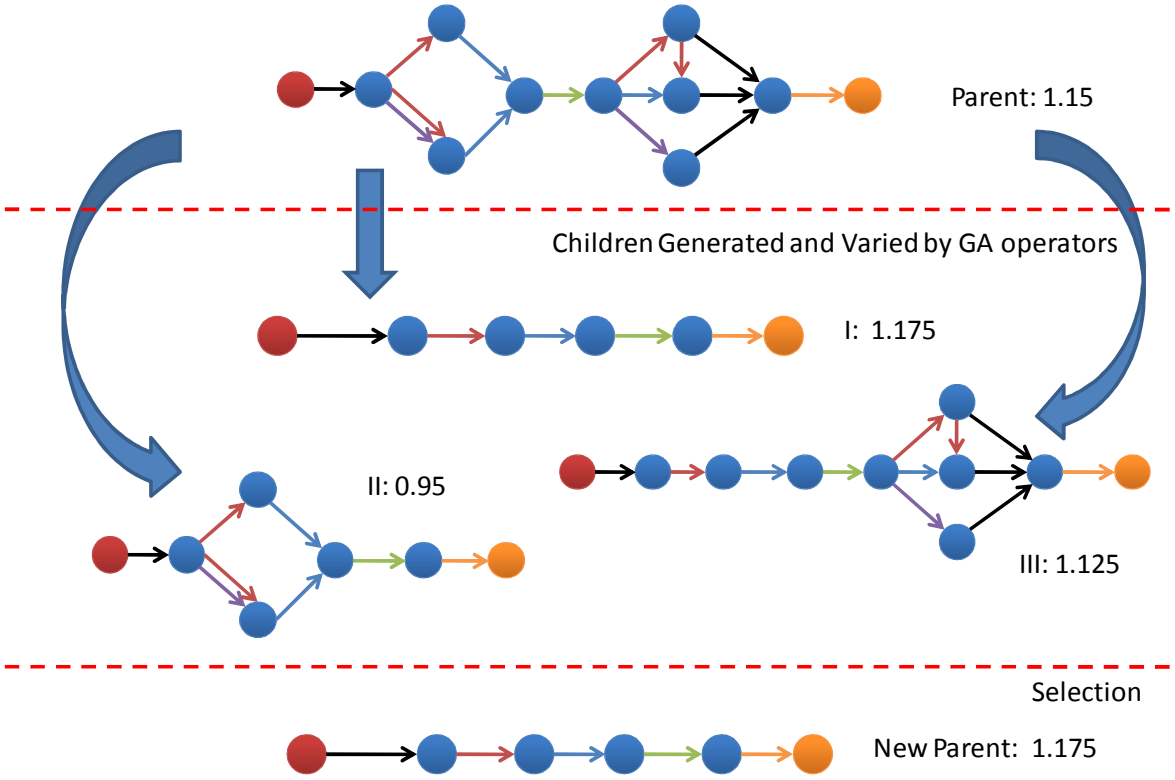


Figure 19. Selection with parameter λ = 3

New Operators for Physical Architecture

As mentioned in the previous chapter, there is a potential need for additional operators to prevent the optimizer from "cheating" and returning unrealistic architectures as optimized answers to the problem presented to it. Also, the new operators described here were designed in order to work with the source and sink vertices and keep them well-tuned and working with the rest of the architecture graph.

The rectification operator is designed to assign unconnected inputs and outputs of component vertices to the environment. The operator creates new edges in order to make these connections. The edge types are restricted to those that are specified within the source and sink vertices. This is a deliberate design decision, meant to demarcate which resources are available in abundance from the chosen environment (working fluids, heat, light, etc.) and which must be generated through technological means (rotational work, combustion). Similarly, the specified physical types of the sink vertex describe the quantities that can be released to the environment.

Since this operator induces a bias toward connections collecting upon the source and sink vertices, a symmetric partner is necessary. The symmetric partner is anti-rectification, which searches the graph for potential places to "off-load" connections from the source and sink vertices. This can be better understood by looking at Figure 20. The operator searches for edges that connect to the source and the sink. Then, it looks for pairs where the tail of the edge connected to the source (the output) is compatible with

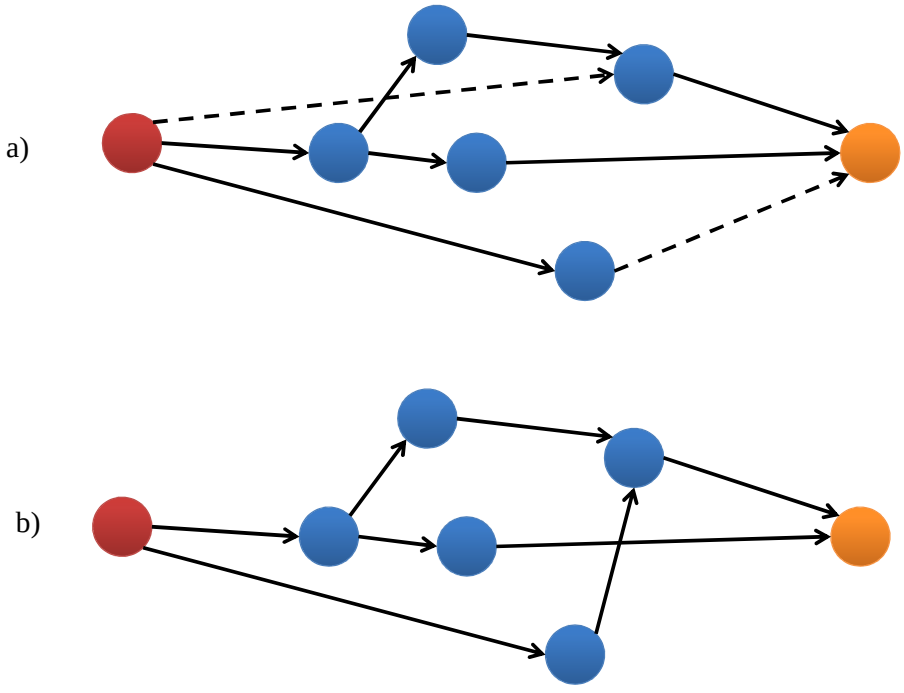


Figure 20. Candidate edge pairs for anti-rectification,, a) in dashed lines and b) moved to new location

the head of the edge connected to the sink (the input). Both of these edges are then deleted, and a new edge is created between the two vertices that were originally connected to the source and the sink.

There is one final operator, the random connection, that simply adds random edges to the graph until all of the inputs and outputs are connected. This is useful for developing initial graphs if there is no pre-determined starting architecture.

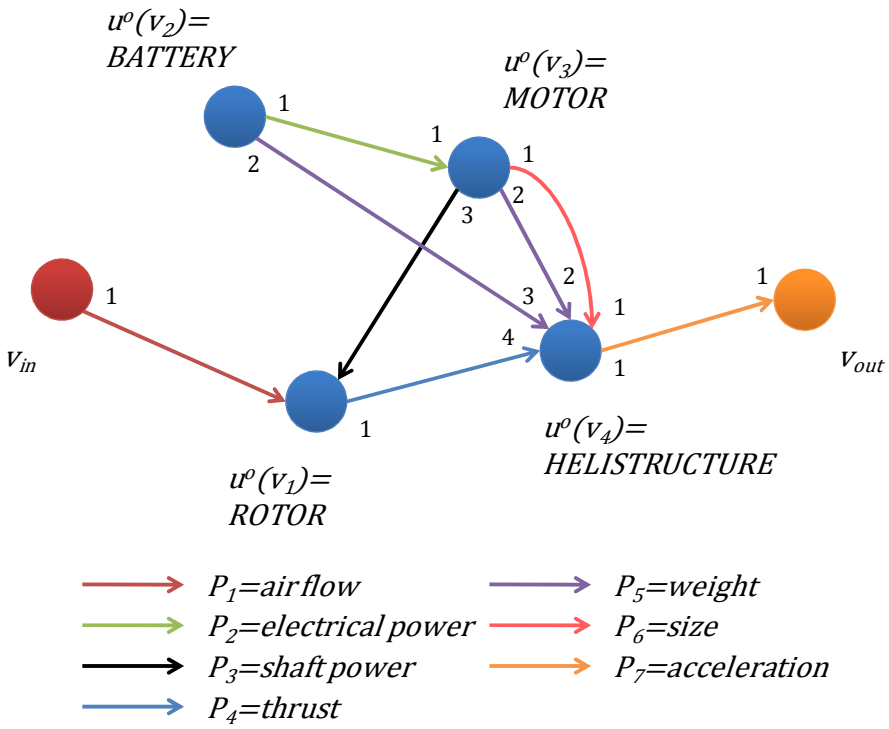
Implementation of Evaluation

The operators described above were conceived of with a graph basis, but are not actually applied directly to a graph. Instead, they are applied to an encoded chromosome, much in the same way a traditional genetic algorithm encodes information on variables into a string. The chromosome is depicted in Figure 21. The first numeral is the vertex type, the second is the destination vertex (the row in the chromosome), the third is the output port, followed by the input port, and finally the type of the edge is represented. These numbers, except for the first, are added on for each edge that leaves a given vertex. In addition to the string that makes up the chromosome, there is information stored within Sindri about the types of vertices represented in the chromosome, the availability of input and output ports for connection, and the types of connections that are considered legal by way of their type.

Thus, up until this point, the discussion of the algorithm has been completely general, and can be applied to any software framework in principle. The main requirement for a software framework to serve as an evaluator in this algorithm is the ability to connect analysis components at will. Programs with this ability include a variety of general analysis frameworks, such as iSight, ModelCenter, Modelica, or Simulink. This ability to create connections must also be done automatically, without a user interface to direct it.

This next section will discuss how evaluations are implemented in Pacelab. In the language of the S'-Graph, the role of the evaluation function is to isolate the elements of the output vector Y of interest, and compose them via the objective function Z into a scalar value.

When the Sindri code is to be executed, the first step is to load all of the appropriate information into the Engineering Workbench (EWB). Engineering Objects for every component to be included in the search, and a source and a sink EO must be placed in the project. If the code is to start with an incumbent architecture, then a series of generic connector Functional Objects should be used to connect the appropriate inputs



Chromosome: (zero-indexed on edges & nodes, one-indexed on ports)
0 3 1 4 3
1 2 1 1 1 3 2 3 5
2 0 3 1 2 3 1 1 5 3 2 2 4
3 5 1 1 6
4 0 1 1 0
5

Figure 21. Chromosome description of S'-Graph

and outputs. The above information is taken into the Sindri code and used to produce an initial chromosome, which is then modified to develop diversity in the population. Also, within the Sindri code itself, the above Engineering Object and Functional Object handles are retained, and combined into a library. This library then forms the basis of the evaluation method.

When the user is setting up the problem, it is done through a dialog screen that allows for multiple configuration options, as seen in Figure 22. The user may currently specify an objective "benefit" as one of the inputs of the sink vertex (usually the first input for simplicity), which will measure the strength of the architected system's interaction with the environment. A cost, which is a variable common to all components such as monetary cost, weight, or size, can be specified as the denominator to an objective function with the benefit as denominator. The specifications will guide the evaluation.

Each time evaluation is called for, Sindri strips away all pre-existing Functional Objects and makes deletes Engineering Objects from the EWB workspace that are not part of the new candidate. It then replicates the Engineering Objects loaded into its library from memory and reinstantiates them within the workspace. FOs are then loaded to connect the components together and complete the graph. At this point, the Pacelab mathematical engine is instructed to solve the graph.

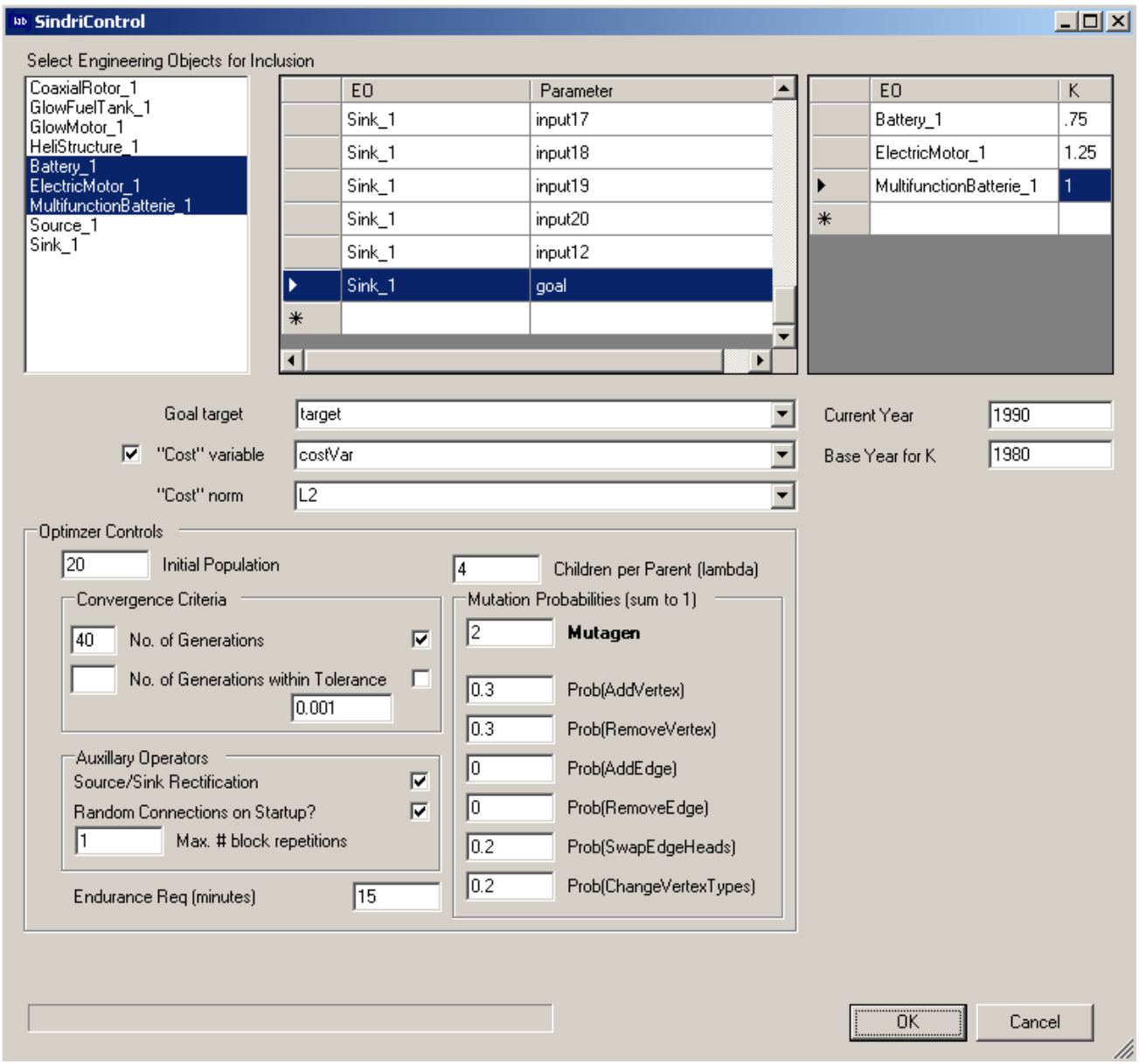


Figure 22. Sindri settings input dialog

Once the graph has been solved, the value of the objective function can be determined. The value of the appropriate input of the sink vertex is queried. All of the components are searched for the value of the parameter with the same name as that the user has specified as a cost. They are then either taken together as the L_1 norm, L_2 norm, or L_∞ norm as requested by the user, and used to form the objective function denominator:

$$L_1(\vec{X}) = \sum X_{i,}, \qquad L_2(\vec{X}) = \sqrt{\sum (X_i)^2}, \qquad L_\infty(\vec{X}) = max(\vec{X})$$

At this point, the candidate solution has been evaluated, and the value can be returned to the genetic algorithm to form the basis of selection.

The problem of evaluation in this context assumed that all of the input values to each of the component vertices have been pre-defined. In the full optimization problem that began this chapter, it is also important to find the input values that lead to the highest objective value for a given set of vertices and connections. This is the subject of the next section.

Continuous Variable Optimization

Up to this point, the combinatorial problem of finding the sets V and E that result in a maximum value of the objective Z, which is composed of outputs Y. The execution of a continuous optimizer for a given V and E serves to find the optimum settings for the input vector $\hat{X}$.

The continuous variable optimization directly utilizes capabilities built into the Pacelab Suite. The package includes support for two different standard techniques, embodied in the CONMIN and NLPQLP optimizers. The NLPQLP optimizer is an implementation of a standard optimizer developed by Schittkowski [138]. The program works by taking a quadratic approximation to the function locally and applying linearized constraints. It is very analogous to Sequential Quadratic Programming (SQP). This approximate subproblem is can be solved rapidly and is used to provide the search direction for the next iteration of optimization. The CONMIN optimizer is also a standard optimizer, built on the method of Vanderplaats [147]. The basis of CONMIN is the Method of Feasible Directions, which determines the direction of the next iteration of search by considering the gradients of both the nearby constraints and the objective function.

Each of these can take into account constraints, which are also standard objects in Pacelab. Further, the full functionality of these optimizers is accessible via the API, and so they can be implemented as optimizers for the problem posed by each graph.

It is also possible for the main objective variable to be designated as a target, rather than a value to be maximized. In this case, there is a somewhat unique feature in the Pacelab environment, due to its mathematical system, that can be utilized. Arbitrary inputs and outputs (the variables that are often calculated internally to an Engineering Object) can have their roles reversed, so long as the number of inputs and outputs remains properly balanced. This is a very useful substitute for situations in which an equality constraint would be desired, which typically place additional burdens on optimizers.

The optimization problem is created automatically by Sindri, via cues built into the Engineering Objects that represent the graph's vertices. Design variables are named with the prefix "design," and the side constraints on these variables are nominated by the "Min" or "Max" suffixes. Technology variables are named with the prefix "techMetric," and are treated as independent variables as well, except for the one dimension (since this is the dependent result of the other values in the multidimensional S-curve). Once a candidate graph has finished with mutation and crossover, the Sindri program scans for independent variables, technology variables, and the appropriate cost variables. These then become the design variables for the optimizer with appropriate side constraints.

To aid the optimizer with what will typically be a multi-modal problem, Sindri also contains a few sample Latin Hypercube tables. These are small, with only ten cases each, but are used as starting points for the optimizer to search for a global optimum. The use of these multiple beginning points also help to make the optimization problem more robust and assure at least one viable solution to submit to the evaluation portion of the genetic algorithm. Also, the API enables a suppress of errors, so failures in the execution of the optimizer do not halt the program.

When the continuous optimizer is employed, the goal is to put the strongest versions of each architectural alternative into the competition. This is also the mechanism through which benefits from technological improvement are made most apparent.

Component Technological Forecasting

With the continuous and discrete optimizers specified, the formulation of the "Find" clause of the S'-Graph optimization problem is complete. It is now time to address the "given" clause in the problem description, which will mark a step away from the optimization algorithm and onto the path to defining a more general method.

In this section, the specification of the technology vector, K, will be addressed. As discussed in the second chapter of this thesis, there are a wide variety of techniques available for forecasting technological development. The purpose of this section is to explain the reasoning that decided which technique was to be used.

The expert survey family of techniques was considered first, due to its popularity, straightforward nature, and use in prestigious studies [15], [14], [109]. However, in the author's experience, such techniques typically revolved around a series of notional technologies currently under development. This did not eliminate survey techniques from consideration, but led to the inclusion of another major family of techniques in the search.

The next family of techniques develop a forecast based on data from the past and extrapolate them into the future based upon a mathematical model. This family is of interest for two major reasons: one, that it brings objectivity to the forecasting process and two, that it has the ability to generate predictions at multiple periods easily. Generating predictions at multiple periods provides maximal flexibility to explore alternative scenarios, such as a doubled or halved rate of progress in various technological characteristics. Also, since the Sindri software will consider multiple

evolving components simultaneously, synchronizing their time scales will be helpful, and again by made possible by the ability to generate values at arbitrary periods. It is also important to note that "objectivity" in this context is not automatically superior to experience and instinct, but is useful in calibrating it.

Three members of this family have been chosen for comparison: the hyperplane [114], the multidimensional S-curve developed by Danner [57], and Technology Forecasting using Data Envelopment Analysis (TFDEA) by Inman [91]. All of them were developed to consider the fact that most systems have more than one performance characteristic of interest to their users, who must often trade one desired trait for another. These trades change their nature over time, which is also accounted for in these models.

While the S-curve and hyperplane methods were straightforward to replicate, the TFDEA process requires multiple steps developed by its original author. However, the Inman thesis contained source code for a TFDEA solver in PHP 4.0. This source code was scanned, put through Optical Character Recognition, and corrected for errors induced by the scanning. One of the support packages, php-glpk [140] was recoded and updated to PHP 5.0 along with TFDEA. Finally, this code was verified by applying it to examples within the Inman thesis and comparing the recoded version's answers to those in the document. Once this was finished, the code was ready for comparison with the other two techniques.

These techniques were evaluated first upon their accuracy. In order to do this, several sets of historical technology data were found and forecast with these techniques. Each of these data sets provides a "home field advantage" for these techniques, as they were used as test cases for publication. The three data sets can be seen in Appendix A. They represent a series of jet fighters (hyperplanes), jet engines (S-curves) and microprocessors (TFDEA).

Once the techniques were used on all of the different data sets, they were analyzed for competitiveness in accuracy. Since the initial results of means and standard

deviations of error appeared to favor the S-curve formulation, null hypotheses were formed for each data set that the other methods performed better on average. Paired t-tests were then applied to the pairs of S-curve-TFDEA and S-curve-hyperplane in order to reject the hypotheses that the S-curve formulation actually had a greater mean error than the other two models.

The results from these tests can be seen in Table 1. The t-test for the absolute deviations are one-tailed, while the test for bias is two-tailed. In all three data sets, the hypothesis that the hyperplane model has lower mean error than the S-curve model is rejected with at least 95% confidence. When it comes to the S-curve vs. TFDEA comparison, the tests are far less conclusive. Only the microprocessor data set shows a clear verdict on the mean error of the two methods relative to each other.

Table 1. Statistical results of attempted predictions of date of introduction

	t-value	p-value
Jet engine data – last 6 values predicted		
$H_0: \mu_{S-curve}^{abs.\ deviation} - \mu_{TFDEA}^{abs.\ deviation} > 0$	1.09	0.34
$H_0: \mu_{S-curve}^{abs.\ deviation} - \mu_{hyperplane}^{abs.\ deviation} > 0$	3.33	0.01
$H_0: \mu_{S-curve}^{deviation} = 0$	1.21	0.28
$H_0: \mu_{TFDEA}^{deviation} = 0$	0.37	0.73
Microchip data – last 14 values predicted		
$H_0: \mu_{S-curve}^{abs.\ deviation} - \mu_{TFDEA}^{abs.\ deviation} > 0$	2.71	0.01
$H_0: \mu_{S-curve}^{abs.\ deviation} - \mu_{hyperplane}^{abs.\ deviation} > 0$	1.79	0.05
$H_0: \mu_{S-curve}^{deviation} = 0$	0.97	0.35
$H_0: \mu_{TFDEA}^{deviation} = 0$	1.11	0.29
Jet fighter data – last 7 values predicted		
$H_0: \mu_{S-curve}^{abs.\ deviation} - \mu_{TFDEA}^{abs.\ deviation} > 0$	0.35	0.37
$H_0: \mu_{S-curve}^{abs.\ deviation} - \mu_{hyperplane}^{abs.\ deviation} > 0$	6.07	0.00
$H_0: \mu_{S-curve}^{deviation} = 0$	1.60	0.16
$H_0: \mu_{TFDEA}^{deviation} = 0$	0.33	0.75

The results above can be better understood by looking further at residual vs. predicted plots for each of the major data sets, which are shown in , , and . In the case of

the microprocessor data set, the residuals are well distributed around the zero line with a random scatter. The S-curve estimates appear to be consistently tighter than those of the hyperplanes, which are tighter than the Inman results. However, a closer look at the residuals by way of normal probability plot shows that the hyperplanes exhibited errors with longer tails than a normal distribution. The other two models showed good agreement with the assumption of normally distributed errors.

The jet fighter data set is different, in that S-curve and hyperplane estimates are both optimistic, and the TFDEA estimate is well-centered. Looking at the residual plot, it appears that this bias is well-spread across the predictions that are made by the two regression-based models. A further check on the residuals was made by way of a normal probability plot, which showed that TFDEA and S-curve residuals were normally distributed, but that the residuals of the hyperplane model were skewed to the right.

The final set, the jet engine data, appeared to present difficulties to all of the models beyond those of the other data sets. However, the S-curve model appeared to have fewer major misses than the TFDEA model, and provide more accurate predictions than the hyperplane model at every point.

From looking at these results, judgments on the accuracy of the methods are as follows. The hyperplane model falls behind in accuracy and appears to have non-normal errors. TFDEA and the S-curve model do not have differences that are statistically significant when examining the means of their performance, the means are relatively small (less than two years when the time unit is integral years), and have normally distributed errors. Thus, from an accuracy standpoint, either of these models are equally likely to perform well on a given problem.

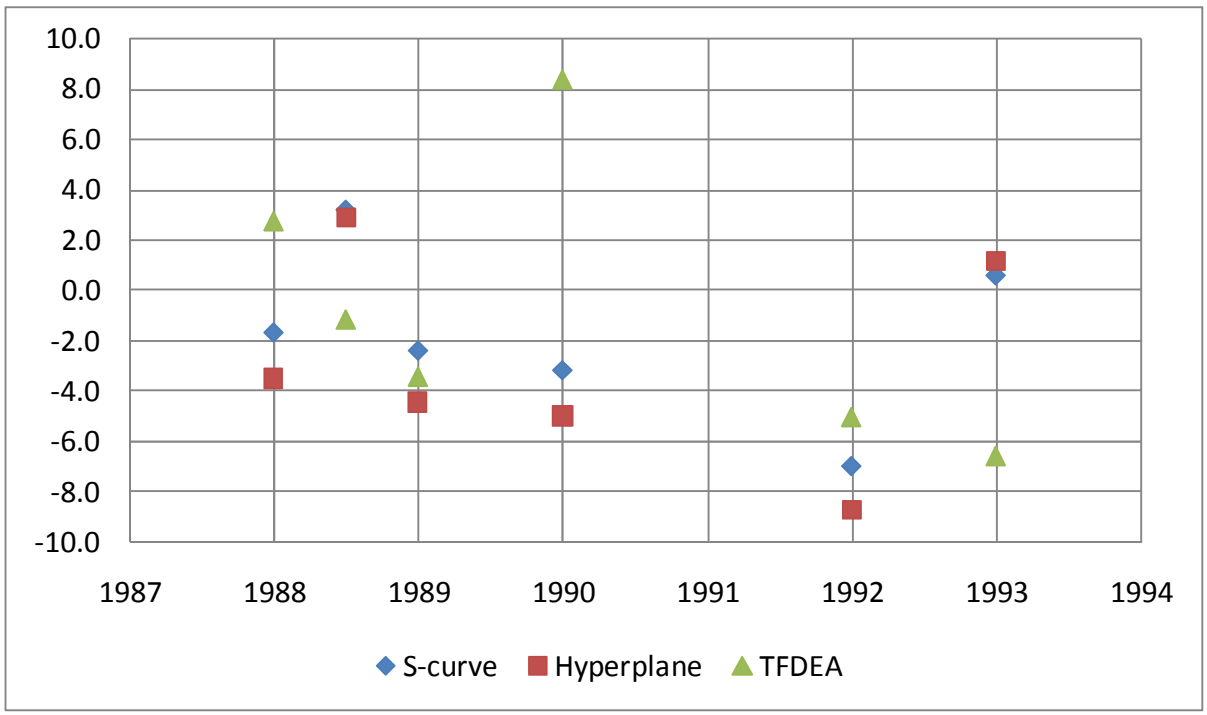


Figure 23. Residual v. predicted plot for jet engine data set

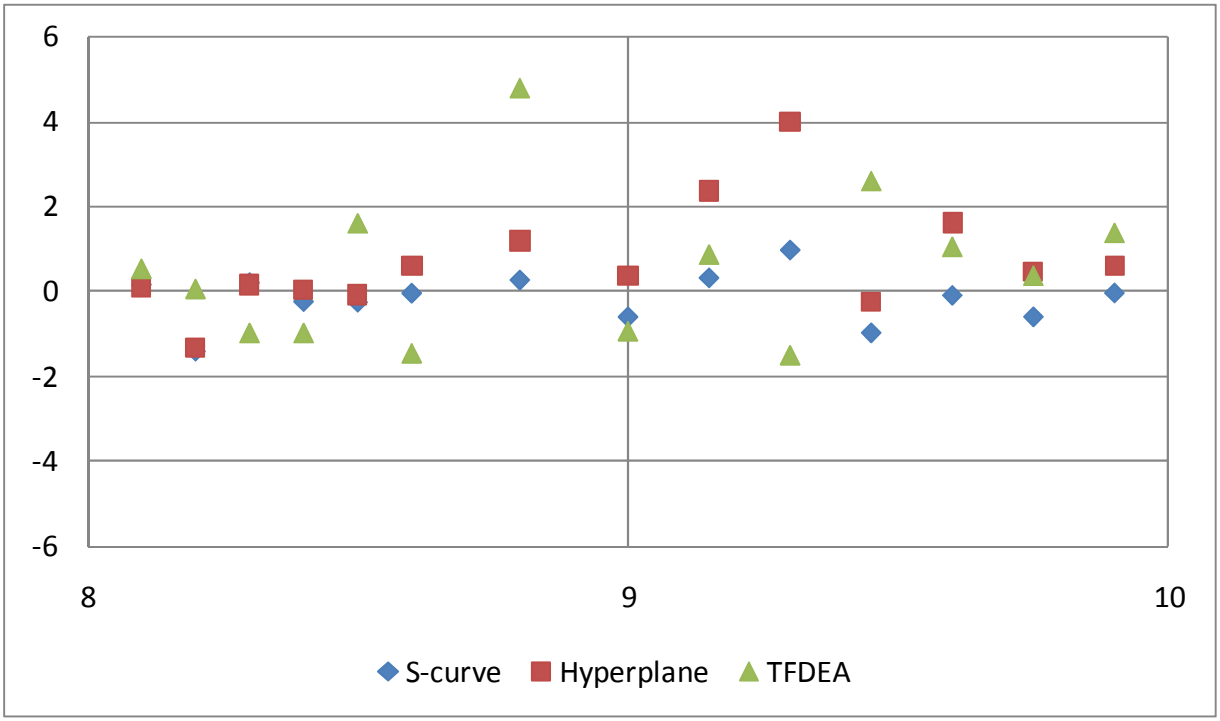


Figure 24. Residual v. predicted plot for microchip data set

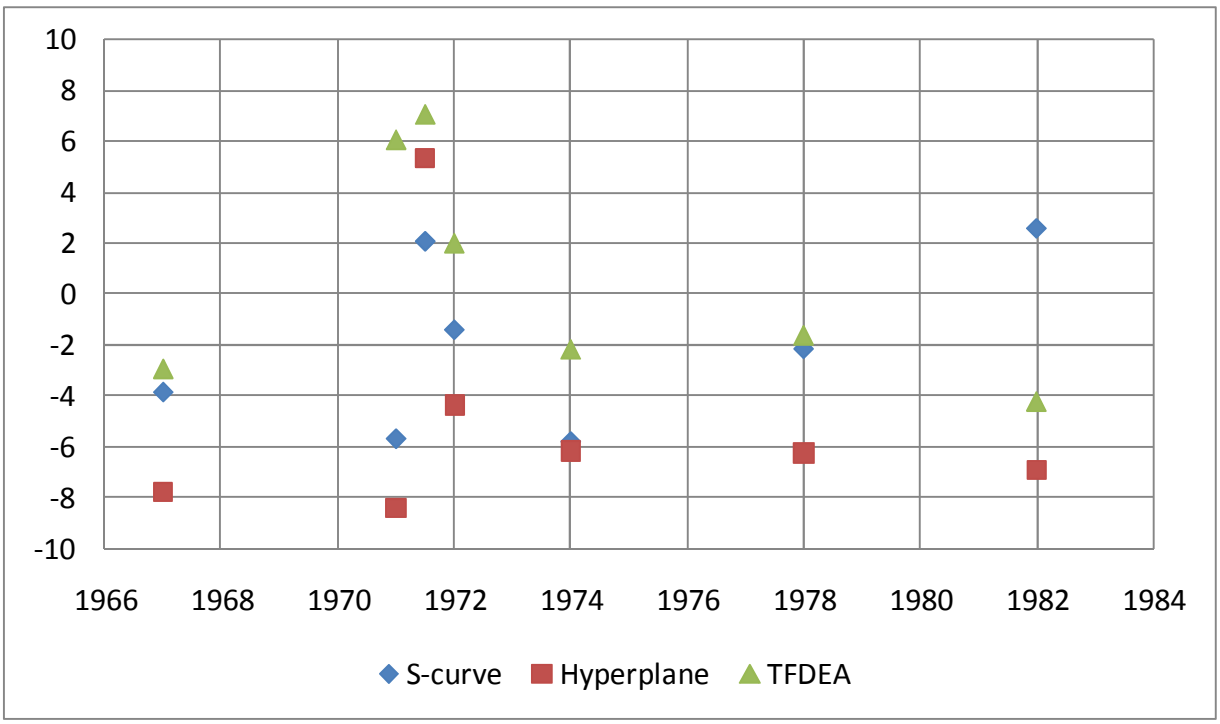


Figure 25. Residual v. predicted plot for jet fighter data set

A quick look at whether the two models are unbiased shows some mixed results. The t-tests undertaken do not show conclusively that there is a bias in the models, but the values are not necessarily encouraging on that front, either. It appears a bit more likely that the TFDEA method is less biased than the multi-dimensional S-curves, but again there is no clear winner. Thus, another potential quantitative discriminator between the two methods is somewhat inconclusive.

At this point, it is necessary to consider other characteristics before choosing a model to use for technology forecasting for use with Sindri. The next characteristics to consider are the simplicity of the model (for programming into Sindri), and the ability to fit these models to the data.

Data fitting is very straightforward with both methods. In the case of TFDEA, the data is fit by the use of a series of linear programs, which are able to rapidly fit data on a computer that can execute PHP and mysql. However, if the computer program is not available, there would be a nontrivial effort to recreate it. As for the S-curve formulation,

it can be developed with either easy to write custom software or commercial statistical packages. For this work, the S-curve is fit by a regression script in MATLAB, which also takes advantage of a constrained gradient optimizer to search for good estimates of saturation limits to the sigmoid curve.

The S-curve model wins the contest on the simplicity of implementation. Recall the variables' transformed state and accompanying model:

$$\mathrm{X_i} = ln\left(\frac{L_i - y_i}{y_i - y_{0,i}}\right), t = \beta_0 + \sum \beta_i \mathrm{X_i}$$

In this form, the result is a simple linear equation with one degree of freedom. With the time set by Sindri, the other technology characteristics can be chosen more or less at will (within the bounds of their ultimate limits) with the final characteristic's value chosen by the formula.

After the comparisons of accuracy over multiple data sets and subsequent considerations of ease of use to serve as the tiebreaker, the multidimensional S-curve can be chosen with some confidence.

Other Inputs to Sindri

The Sindri code, which solves the "find" clause of the S'-Graph problem, takes as inputs all of the quantities described in the "given" clause. The discussion on technology forecasting in the previous section addressed the vector K. A full account of the formulation of $\tilde{V}$, O, and P are still required. Each of these mathematical objects can be given a specification through the use of engineering judgment and common practice – there is no need to develop specialized techniques for them.

The $\tilde{V}$ object simply represents all of the components that are to be taken into consideration when architectures are composed. Thus, it is up to the one employing this method to research the components of interest to the problem and to select those that are considered relevant. In the example of the MAST program, the $\tilde{V}$ object would likely

include every subsystem of interest, including modes of locomotion, sensors, power sources, and computing resources.

Once these components have been decided upon, the next step is to develop analytical models. The analytical models are represented in the S'-Graph problem as the object O, which transforms X and K into the outputs Y. These models can be developed in any way that is found to be fitting and at any desired fidelity in theory. Practically speaking, it is recommended that the models be developed with a fidelity consistent with conceptual engineering, namely that there is "an outline solution … worked out in detail for it to be possible to supply approximate … weights, and overall dimensions" [78] in addition to approximate levels of performance. This will require the modeler to employ empirical factors or estimates of future performance, several of which would be good candidates for addition to the K object and forecast over time.

The final object to be specified is the set of Sindri tags, or labels for inputs and outputs to represent the physical interfaces of components. This is the object P in the formal problem. As with the modeling, this is a matter of engineering judgment. The physics of the component under use will usually be a good guide to the appropriate tags, such as energy, work, airflow, weight, and so on.

Using the Outputs of Sindri

The Sindri code finds an optimal architecture for a given set of conditions, after the architecture has been formally specified in the S'-Graph form. The solution to this problem is based upon a number of assumptions and quantitative inputs. This solution can be used as a point solution to a problem of optimization. However, since Sindri is an automated code, it can be leveraged to get a much richer strategic picture by investigating the sensitivity of the solution to the assumptions that are made to operate it.

For example, consider again the use case of an incumbent architecture being pitted against a number of candidate successors. This problem is affected by the rate of

progress made on the technologies that enable the potential successors to be competitive. In this process, the baseline rate of technological advance has been encapsulated in a multidimensional S-curve. This rate can be modified by accelerating or decelerating the progress of time to emulate technologies emerge ahead of or behind schedule. In this case, a k-factor (not to be confused with the vector of technology characteristics K), can be applied to the time entered into the formulation for the multi-dimensional S-curve:

$$t_{effective} = k(t_{current} - t_{base}) + t_{base}$$

The new effective time is then used in the S-curve formulation in place of the original time. In the above equation, the "current" time is the time to which the forecast is applied. For example, a forecast for the year 2020 might have a base year of 2010, and a k-factor of 1.5 applied, meaning that the technology predicted for 2025 arrives five years early.

Visualization

The visualization regime for Sindri's output is designed to focus on the original problem laid out in this document: to determine when it is likely for an ascendant architecture to become competitive. Thus, the visualization must clearly show where the potential transition points occur, but also convey a sense of sensitivity to the estimates. This indicates that a kind of phase plot (phase is in state of matter, not frequency domain) would be an appropriate element for visualization.

An example is illustrated in Figure 26. This plot would be replicated over the total number of technology metrics of interest, which could be as the sum of the number of metrics of all of the components under consideration. The horizontal axis represents time, starting at some baseline of interest. The vertical axis represents a multiplier upon the rate of development, with unity indicating the rate indicated by the appropriate multidimensional S-curve. The non-unity rates indicator multipliers of the time taken from the baseline start time. For example, on the plot labeled with Technology Metric 1 with a baseline of 1990, the point of (1996, 0.5) would indicate that the data were run with all technologies evaluated at the 1996 level, except for Technology Metric 1, which is at the 1993 level. Thus, a basic sensitivity of the results to this metric's rate of improvement would be shown.

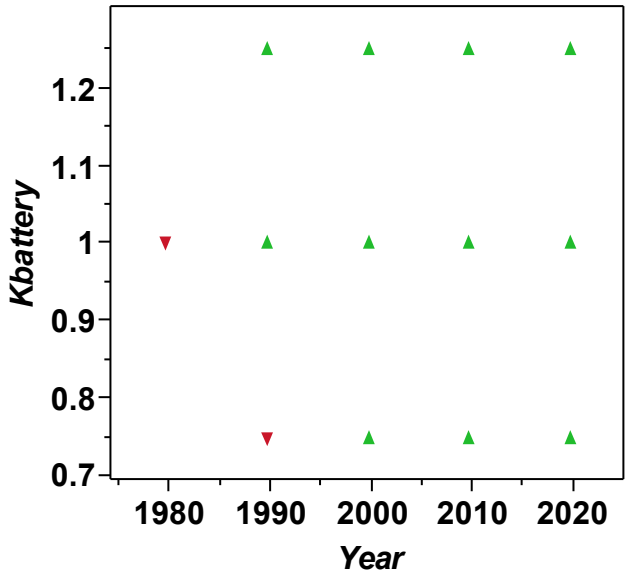


Figure 26. Two-dimensional progress sensitivity plot

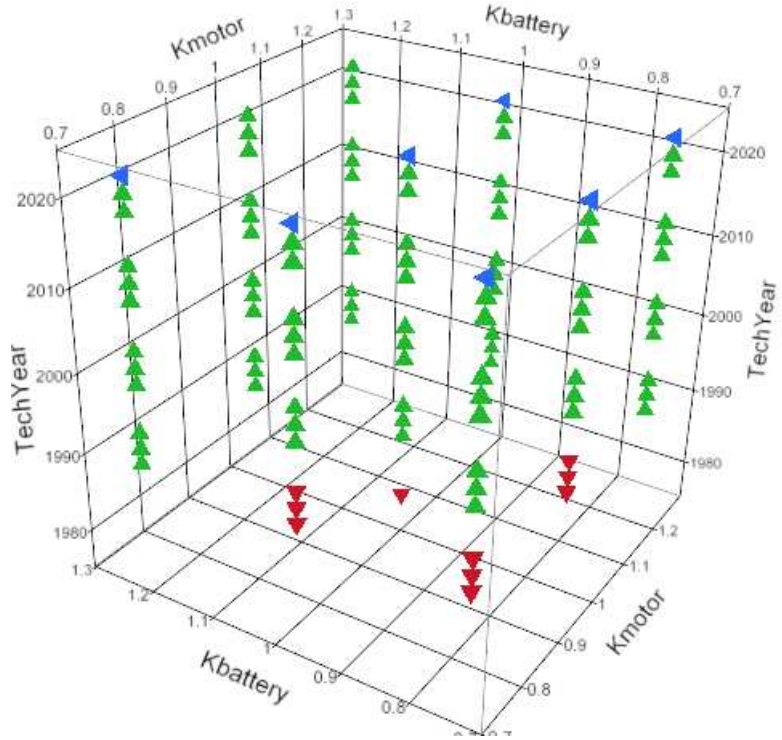


Figure 27. Pseudo-four-dimensional plot

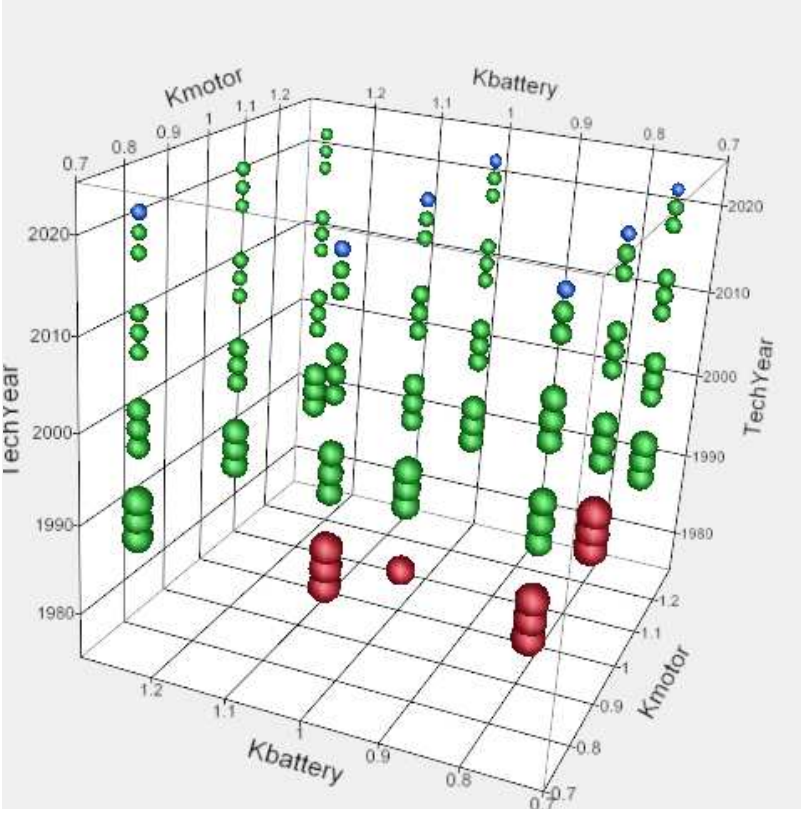


Figure 28. Pseudo-four-dimensional plot with a scalar response

In the case of lower-dimension (less than six dimensions) spaces, there are two plots that can be very useful for visualizing transitions between architectures and their sensitivities. The first is a plot of architectures, using different symbols, in three-dimensional space. Two of the dimensions can represent sensitivities, while the third dimension represents time. Since this is a series of discrete points, it is possible to gain limited insight into the third, fourth, and fifth sensitivities by clustering the new data around a central point. As long as the points within the cluster are closer together than the clusters, it is possible to discern the differences in dimensions. This is a limited way to see more than three dimensions simultaneously. An example is shown in Figure 27.

Further, it is possible to use this technique to represent both discrete outcomes (such as the type of architecture being represented by symbols) and continuous ones (such as an object the architectures are designed to reach). An example of this is shown in Figure 28.

Walkthrough of Method

All of the components of the Sindri code and supporting techniques have been described above, and come together into a single method. The method is designed to solve an initial problem of wanting to understand the potential shifting points between architectures. The flow of this method is illustrated in Figure 29. There are a series of discrete steps, which are reviewed below.

Step 1. Specify Problem

This is the quintessential first step for any analytical method. A good problem definition for the Brokk method requires the types of systems of interest, the components to build up these systems, a timeframe of interest, and the currents of technological advance to be considered. Even though candidate architectures are to be developed by the Sindri algorithm, the ability to properly formulate components can be aided by having

a mental framework to work within. For example, even though there are no dominant architectures in MAST, there is certainly no shortage of ideas. Scorpions, dragonflies, spiders, bats, and sand crabs all have analogues in the project. DARPA's Micro and Nano Air Vehicle [86] projects can also contribute ideas for system architectures.

Step 2. Specify Component Models.

The method proceeds with some background work in order to develop a series of models for the components that will be combined to form candidate architectures. The components must be defined by an analytical model to relate inputs to outputs. This can be done with existing conceptual and preliminary phase design practice. Typically during a conceptual design, the model is greatly simplified, and relies heavily upon a series of empirical performance factors or historical constraints. For aircraft, examples of this would be jet engine specific fuel consumption, feasible thrust to weight ratios, feasible wing aspect ratios.

The component model step also requires the user to specify the physical inputs and outputs from the component. This is to be loaded into Sindri via the tagging construct.

Step 3. Identify Technology Factors

These empirical factors that reveal themselves during the modeling process are candidates to become technology factors. At this point, it is up to the practitioner to decide which of these candidates are important, as well as likely to evolve in the future. Once the list of technology parameters has been established, the next step is to develop extrapolation models for their advance. In this case, the model to use is the multidimensional S-curve, as established earlier in this dissertation.

Step 4. Fit Multi-dimensional S-curves to Technology Data

Once the technology factors have been identified, the question of projection into

the future must be answered. This may be done with either multi-dimensional or single-dimensional S-curves. A fitting procedure for the former has been encoded in MATLAB, and can be seen in Appendix C. It is up to the user to determine how best to check the goodness of a given fit.

Step 5. Execute Sindri Model over Times of Interest

Once both component analysis and technology evolution models have been developed and verified to satisfaction, they must be loaded into Sindri. The year and the K factors (which accelerate or decelerate the appropriate technological component's progress) must then be entered. The analysis models should be constructed as Engineering Objects in Pacelab, and then loaded into a new project. Executing the Sindri code, as shown in the figure, will recombine and link the components together via genetic operators. The continuous optimization problem will be automatically built using the parameters with "design" in their names and constrained according to the "design(Name)Min" and "design(Name)Max" patterns.

Step 6. Build Results Database

Once an optimal architecture has been found for the defined year, rates of technological progress, and component analyses, it can be added to a database. This architecture then becomes a data point, with the various factors of technology and time. At this time, it is recommended to use a full factorial design based upon having a sensitivity factor for every component with a technological forecast upon its parameters. This will enable the consideration of "what if" games in the visualization step.

Step 7. Visualize Results

Once a database of times, sensitivity factors, and architectural outcomes has been built, it can be visualized according to the techniques laid out in this chapter. At this point, the data will be available for the user of this method to gain a better understanding

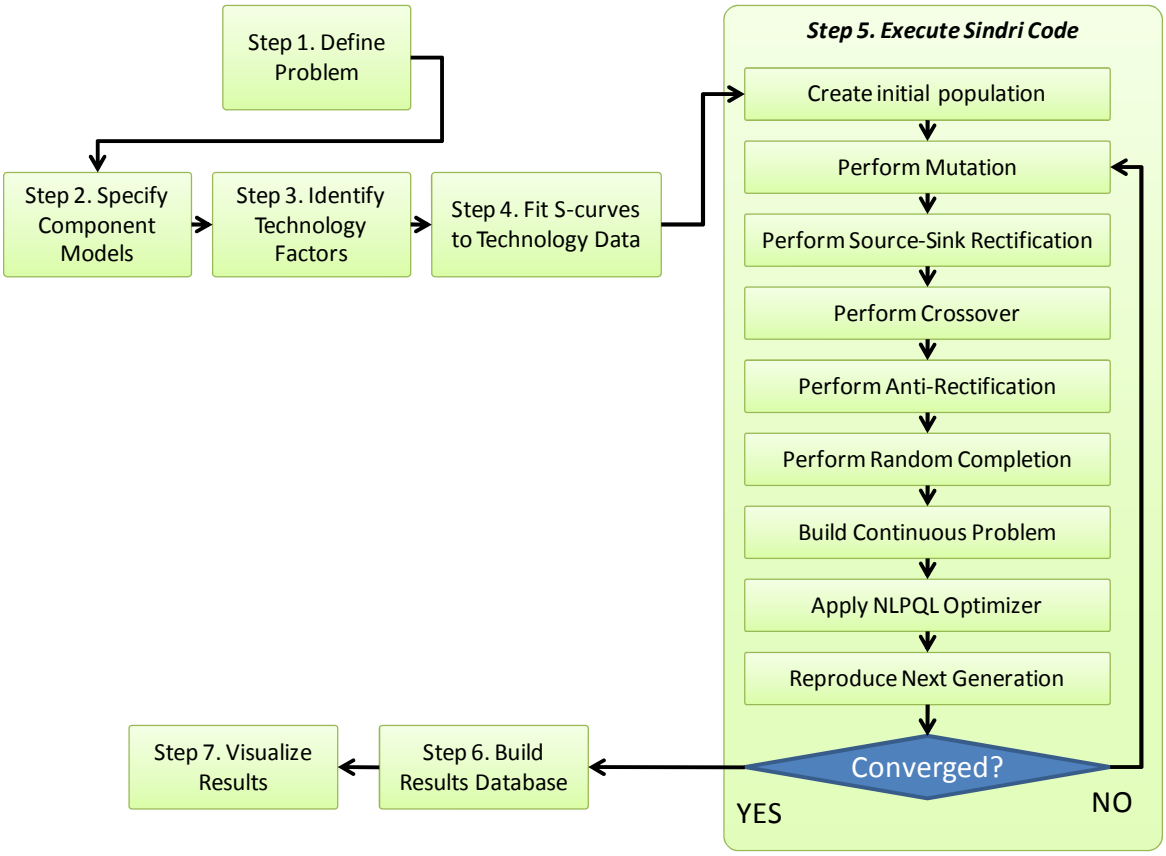


Figure 29. Brokk method flowchart

of when and how architectural shifts may come about in the near future. The visualization should be used to look for frontiers and borders where architectural outcomes change.

The visualization techniques just discussed in the previous section are a useful way to search for these frontiers in many dimensions simultaneously. Since the database described in this method is built from discrete points and a full factorial design, the data can be viewed with minimal post-processing. Also, more traditional plots of performance or cost characteristics changing over time may be of use to more thoroughly understand the results.

CHAPTER 5

INVESTIGATION

With a general method for finding transition points between architectures that is sensitive to time and technology, the time has come to investigate its performance. The parameters of interest in its performance were laid out when hypotheses were formulated in previous chapters. These hypotheses represent the point at which thought and theory could be applied no further, and it became necessary to perform tests in order to make comments upon the method's performance.

There are two arenas of investigation discussed in this chapter. The first is a test of the genetic algorithm described in the previous chapter and the subject of many of the hypotheses considered with this dissertation. In order to explore the operation of the algorithm, a set of test combinatorial spaces were generated. The second arena of investigation is a test case that is designed to demonstrate how the Sindri code and its supporting techniques would forecast architectural transitions. The test case is designed to help collect the first set of data in supporting Hypothesis One, regarding the ability of a genetic algorithm to find an optimal architecture. The test case develops data in order to forecast the transition points between different architectures. This further provides an opportunity to display not only the baseline forecast, but also the sensitivity of the forecast to deviations from the expected rates of progress.

The goal of the combinatorial space is to facilitate an isolated investigation of the performance of the genetic algorithm within Sindri, and to build a base of evidence for the support or rejection of hypotheses based on its performance. The space provides a computer experimental apparatus that can be used with a program of run settings to systematically develop performance data to be used as supporting evidence. The systematic collection also allows for the confident use of standard analysis techniques

and statistics in order to form balanced judgments about the results. Further, this design space may be used for future scholars to use to evaluate competing techniques.

Once the data from executing the genetic algorithm within this space was collected, it was used to make minor modifications to the methods built into Sindri. This included changing the order in which different operators were performed, and the introduction of random completion into the main execution loop. At first, the random completion was an auxiliary operation, meant only to generate architectures when the user declined to specify a baseline. By using the completion within the loop, the open ports are forced into a connection in order to get more realistic architectures. It also became apparent that the addition and removal of edges within the mutation operator was not worthwhile. Swapping the heads of the edges is still used in order to attempt to find the optimal set of edges for a given set of vertices.

After these adjustments were made, the test case was performed. In this case, the system to examine was a micro rotary air vehicle. The goal was to capture the transition between glow fueled models and electrically powered ones, and then to make an example forecast of the transition to multifunctional and biologically inspired structures.

Combinatorial Test Space Generation

The performance of an optimization algorithm is typically evaluated in two major ways. The first is mathematical considerations, such as proofs of convergence on various types of spaces. The second is to test the algorithm on a simplified test problem where the optimum, and possibly certain properties of that problem, are known. An example of this is Rosenbrock's valley function [134], which challenges gradient optimizers with a long, curved, and relatively flat valley. Another example is the "egg crate" function, which is a sinusoidal function intended to test the ability of optimizers to find a global optimum among many local optima.

Once this search space is constructed, it will become possible to get some sense of

its topology and identify features that make it difficult to traverse. Pointing to these aspects would be the same as highlighting the difficult valley or multiplicity of local optima described in the Rosenbrock valley and egg crate functions. In addition, the optimum value can be obtained exactly and compared to the result of the optimizer. The rate of approach to this optimum value can also be tracked.

To properly formulate a useful test space for the genetic algorithm implemented in Sindri, it is important to recall the problem type. Not only is the optimization problem a combinatorial problem, but it also involves the construction of a viable architecture represented by an S'-Graph. The test space will thus require some number of simple and standard "components," as well as a logic to connect them. In addition to the standard components, a number of standardized connection types will be required to emulate the physical connection types described in the previous chapter. The connection types will be used to label the inputs and outputs of each component and restrict the ways in which they can connect to one another. In the case of these examples, the somewhat frivolous connector types of chocolate, vanilla, and strawberry were created.

A further consideration for this search space is the fact that there are a number of feasible solutions that correspond to a completed architectures in a vast background of non-completed architectures. These non-completed architectures will have some number of connections, but certain input and output ports of the vertices will be left open. When these architectures are evaluated, these open ports would possibly resemble some non-physical result (e.g., a jet engine producing thrust and ingesting air, but not consuming fuel). Thus, these non-completed architectures must be excluded from the search space, with the assumption that they would also be excluded when it came time to evaluate them.

With these considerations in mind, it is possible to design a computer program to create all of the possible, completed architectures that are generated by combinations of a given set of component vertices. This program was crafted in the form of an Excel

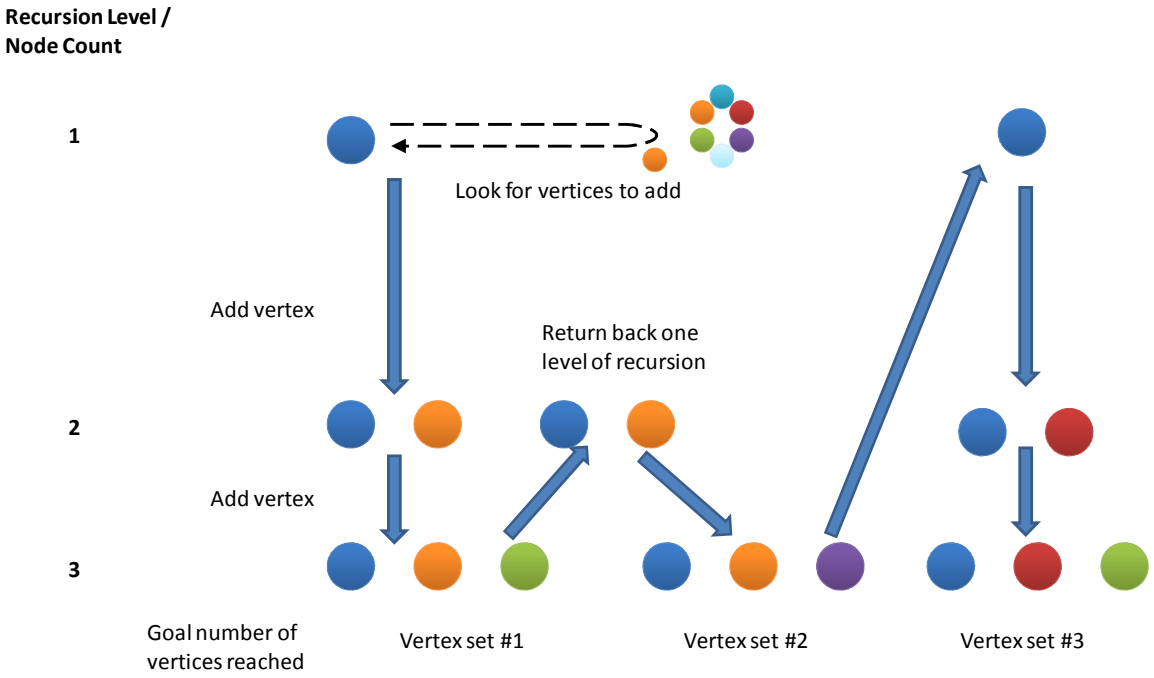


Figure 30. Generation of graph vertices by combinatorial space creator

macro, and is listed in Appendix B. To some degree, it mimics the connection logic used in Sindri, in that it looks for unused connections and labels of inputs and outputs to each node. The macro takes in a list of vertex descriptions and a limit on the number of times each vertex can be used. The macro then searches for graphs of one vertex, two vertices, and so on until the upper limit of possible vertices is reached.

The search for vertex combinations is performed through a recursive routine, which is illustrated in Figure 30. The routine starts by adding a vertex from an available pool. When the new vertex is added, the next step in recursion is taken by adding another new vertex. This is done until the graph is at its full length. The graph is then given to another routine that is responsible for building up all of the possible connections in the graph. Once this is finished, the recursion then begins to unwind itself by stepping backwards. At each step backwards, however, there is a counter that pushes the macro to add a vertex of a different type than it had added in the previous step through the recursion. Thus, the recursion builds out all of the combinations of the graph, starting by varying the final vertices, and moving backward until all options are explored.

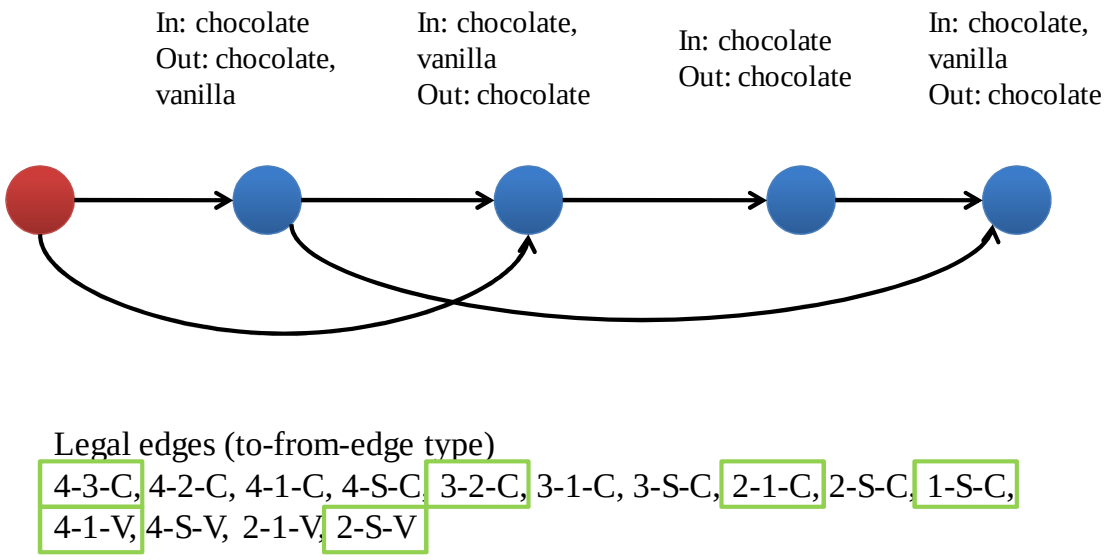


Figure 31. Generation of graph edges by combinatorial space creator

When each graph is complete, an edge generation routine develops a list of all possible edges, taking into account the limits on compatibility implied by having different edge and port types. Then, as with the vertices, a recursive subroutine branches from each newly added edge until the graph is as well-connected as possible. If the graph is not well-connected, meaning that only one output port is unused (presumably to be tied to the sink), it is rejected and not added to the list of viable combinations. An illustration of a graph that is built this way, with the possible edge list and the edges that were selected, is given in Figure 31.

It turns out that in many cases, a given set of vertices will not have a single viable graph because the possible in-degree (number of input ports) of a given type is not equal to the out-degree. In this case, the macro removes the list of vertices as well as not generating the graph. If a given set of vertices gives rise to multiple viable combinations of edges, each set of edges will be matched with a duplicate of the vertex set.

It should also be noted that this routine works by moving from left (source side) to right (sink side) on the vertices, as depicted in Figure 31. This is possible because the search space is limited to directed acyclic graphs, and so the vertices can be ordered. Finally, although only two different input/output "tags" were used, three-tag variants of

the search space were developed, and so there is some reason to believe that the algorithm would work in general.

Once all of the possible graphs have been generated within the macro, the macro automatically generates spreadsheet functions within Excel. This allows for the user of this macro to tweak the parameters of the various functions represented by the vertices and tune the space. It should also be noted that these parameters are tuned in specifying the test space, but they are not treated as design variables when testing the genetic algorithm.

By executing the macro, a combinatorial design space is generated and every member of that space is evaluated. At this point, the components were specified as they would be in Step 2 of the Brokk method. A series of test functions were devised and transformed into Engineering Objects for the Pacelab environment. The building blocks of the test design space are thus loaded into the Sindri code and used to test the characteristics of the genetic algorithm built within it. The genetic algorithm of the Sindri code is then able to construct solutions within the design space and evaluate them. In this test case, the continuous space optimizer is not used – the Pacelab environment is simply called upon to evaluate the solution graphs once they are developed.

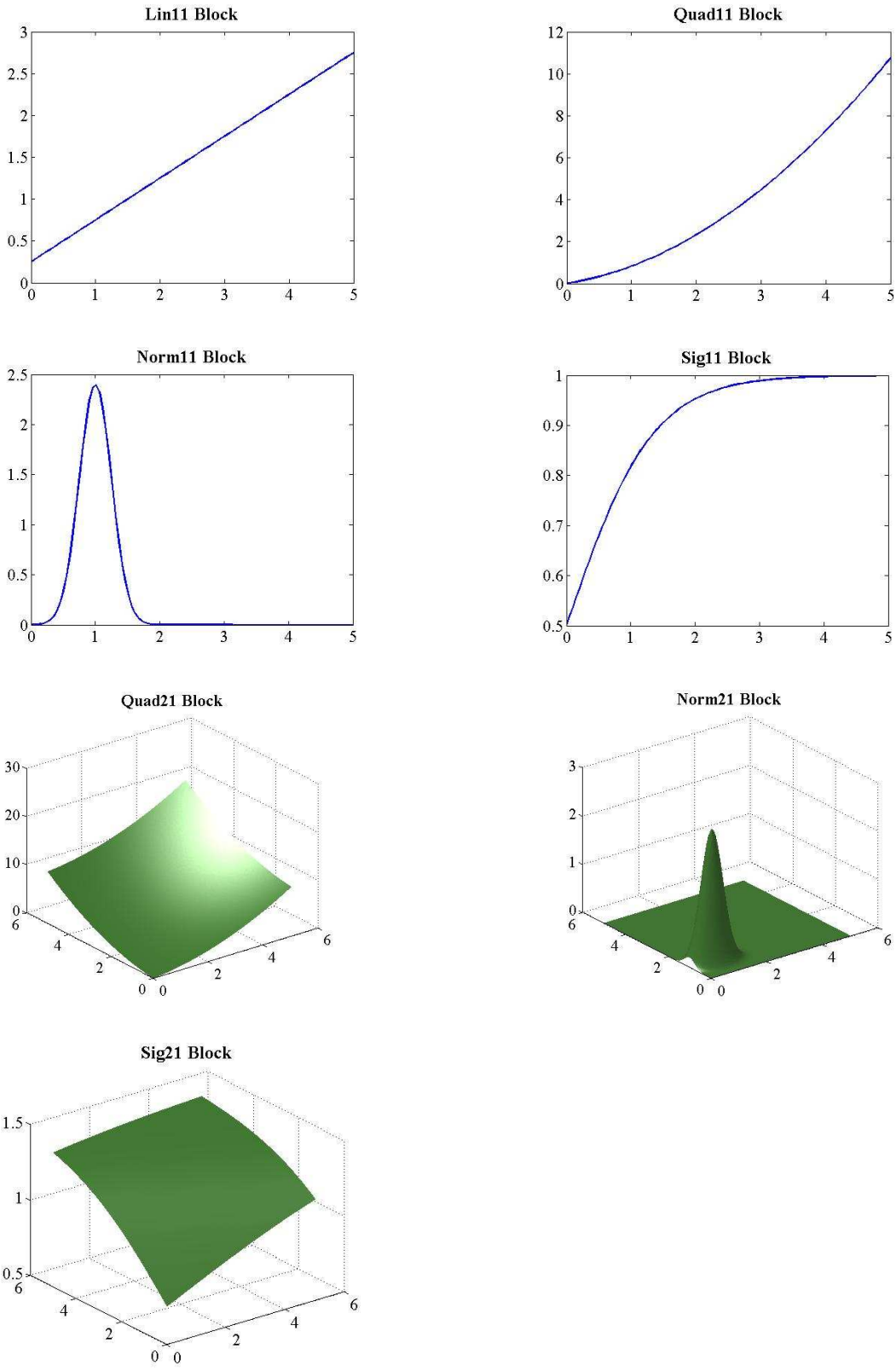


Figure 32. Simple functions used in combinatorial test space

The individual test functions are very simple functions, both for computational speed during the trials and for the ease of verifying and programming. The functions represented very basic input/output maps. They were: linear, quadratic, normal, and sigmoid functions. The linear and quadratic functions were meant to represent increasing functions of varying strengths, while the normal function represents a target, and the sigmoid represents a saturation characteristic. Thus functions were chosen to present a challenge to the genetic algorithm, in that each of the above function types has an optimal role in an architecture. Since the quadratic and linear blocks are unconstrained in their limits, they are often best place nearest the sink, providing a final boost to the output value. A sigmoid function can provide a quick gain in the output, but quickly reaches a limit. And finally, the normal function often serves to "poison" the chain by mapping a high input value into a low output.

Each of these functions was meant to have their design variables left frozen, and simply perform the functions shown in Figure 32. This left the genetic algorithm to concentrate on the combinatorial space represented by vertices and their potential connections. The task presented to the genetic algorithm, in other words, was to develop a graph that maximally leveraged the properties of the functions that each vertex represented. There are one-dimensional and two-dimensional functions. In the case of two-dimensional input and one-dimensional output, the functions were joint, also shown in Figure 32. One-dimensional input and two-dimensional output functions are obtained simply by having two of the one-dimensional functions in the same package and sharing the same input variable value.

Ultimately, three test search spaces were created, mostly constrained by time. The three test spaces, and their characteristics, are shown in Table 5. It should be noted that there are many, many more graphs than are feasible networks by the definition used here (all vertex output ports in the graph used except one). For example, the number of potential connections in the first design space is equal to the number of labeled graphs

with 8 vertices, or just under 270 million. The time constraints came in part from the number of trials made on each group of settings, but was mostly due to the need to debug Sindri. The Design of Experiments did at least one of its jobs very effectively, which was to put the Sindri code into almost every possible state it could be in. Every bug in indexing, miscopied chromosome strings, or other subtle error was exploited to crash the program. The uncanny ability of the DoE to crash the program, needless to say, greatly increased the time required to complete the data collection.

Table 2. Test combinatorial search spaces

	Space I	Space II	Space III
Functions Used	Lin11, Quad11, Norm11, Sig11	Lin11, Quad11, Norm11, Sig11	Lin11, Quad11, Norm11, Sig11, Quad21, Norm21, Sig21, Lin1Norm1, Quad1Sig1
Weighted?	No	Yes	No
Feasible combinations	7364	7364	142257
Number of repeats of a block allowed	2	2	1

Experimental Design

Hypotheses 1 through 3 are all statements about the performance of the genetic algorithm portion of Sindri under various conditions. In order to make the tests of the algorithm as systematic as possible, a Design of Experiments was conceived. Since it was possible that large numbers of functional calls would be required for each group of settings, the size of the DoE was kept small. The design was meant to be a screening

design, to search for the relative size of the main effects of altering mutation probabilities and the use of random starts and source-sink rectification.

The parameters to be altered in the Design of Experiments are meant to capture all of the aspects of the genetic algorithm. Source-sink rectification (and its partner) and random completion of graphs are both non-standard operators for a genetic algorithm, and so the ability to turn them on and off are useful. Population size and the number of children every member creates per generation interact with the number of generations used to reach a solution to generate the number of function calls the algorithm makes. Overly large populations can be wasteful, while populations that are too small may not have enough diversity to reach a good solution. The mutagen parameter determines how many mutations a given member of the population is likely to experience, and thus determines how rapidly the algorithm can move through the design space by mutation alone. Finally, since the mutation operator is itself composed of multiple types of changes to a given member of the population, it was thought to be worthwhile to see how much tuning could be applied.

The design is given in Table 3 for the one-type design spaces, and in Table 4 for the two-type design space.

Table 3. Design of Experiments for Design Spaces I & II

Rectification	Random Start	Initial Pop.	λ	Mutagen	Add Vertex	Add Edge	Swap Vertex	Swap Edge
L1	L1	15	4	6	0.2	0.2	0.2	0.4
L2	L2	15	2	6	0.4	0.2	0.2	0.2
L1	L1	15	2	2	0.4	0.2	0.2	0.2
L1	L1	15	4	2	0.2	0.2	0.2	0.4
L1	L2	40	2	6	0.4	0.2	0.2	0.2
L1	L2	40	2	2	0.2	0.2	0.2	0.4
L2	L1	15	4	6	0.2	0.2	0.4	0.2
L2	L2	40	4	2	0.2	0.2	0.4	0.2
L1	L2	15	4	6	0.2	0.4	0.2	0.2
L1	L1	40	4	6	0.4	0.2	0.2	0.2
L2	L2	15	4	2	0.4	0.2	0.2	0.2

L2	L2	40	4	6	0.2	0.2	0.2	0.4
L1	L1	40	2	6	0.2	0.2	0.4	0.2
L1	L1	40	4	2	0.2	0.4	0.2	0.2
L1	L2	15	2	2	0.2	0.2	0.4	0.2
L1	L2	15	2	6	0.2	0.4	0.2	0.2
L2	L1	15	2	6	0.2	0.2	0.2	0.4
L2	L1	40	2	6	0.2	0.4	0.2	0.2
L2	L1	40	2	2	0.4	0.2	0.2	0.2

Table 4. Design of Experiments for Design Space III

Rectifi-cation	Random Start	Initial Pop.	λ	Mutagen	Add Vertex	Add Edge	Swap Vertex	Swap Edge
L2	L1	80	8	2	0.2	0.6	0.1	0.1
L2	L1	80	8	8	0.05	0.6	0.3	0.05
L2	L2	200	8	8	0.2	0.6	0.1	0.1
L1	L2	200	8	8	0.05	0.4	0.5	0.05
L1	L1	200	8	2	0.05	0.6	0.25	0.1
L1	L1	200	2	8	0.05	0.6	0.25	0.1
L2	L1	200	2	8	0.2	0.25	0.5	0.05
L2	L2	200	8	2	0.2	0.25	0.5	0.05
L2	L2	80	2	8	0.05	0.6	0.25	0.1
L1	L1	80	8	8	0.2	0.6	0.15	0.05
L1	L2	80	8	2	0.05	0.35	0.5	0.1
L1	L1	80	2	8	0.2	0.2	0.5	0.1
L1	L2	80	8	8	0.2	0.2	0.5	0.1
L1	L1	200	8	2	0.05	0.4	0.5	0.05
L2	L1	200	8	8	0.2	0.2	0.5	0.1
L2	L1	200	2	2	0.05	0.35	0.5	0.1
L1	L2	200	2	2	0.2	0.6	0.1	0.1
L1	L2	200	2	8	0.2	0.6	0.15	0.05
L2	L2	80	2	2	0.05	0.4	0.5	0.05

The design was created through the statistical software package JMP as a custom design. The main reason for a custom design was the fact that the mutation type (e.g., add/subtract node) probabilities were required to add to 1, and so required consideration as a mixture. This type of experimental design is generally known as a mixture-process design. The parameters of population size, children parameter λ, and mutagen were continuous variables, while the switches for a random start and the use of source-sink

rectification were taken as ordinal variables. Population size was a difficult parameter to specify, since it required some understanding of how well the population would cover a given search space. It was essentially determined by trial and error on some test versions of the search spaces.

Due to the difficulty generating a large number of runs due to troubles with debugging Sindri, a D-Optimal design was chosen. JMP's internal variance tools were used to consider how many design points to use. The first, the Prediction Variance Profile pictured in Figure 33, was used to evaluate the balance of the design, and to evaluate which of the effects would be seen the most clearly after the design was executed. In the design that was chosen, the rectification and random start variables had minimal variance, population, mutagen, and lambda had slightly more, and the mutation operator variables had the most, which was biased toward higher settings having lower variance.

The value of the variance on the Y-axis is a relative variance. In the description of this type of plot [28], it is said that this variance should be multiplied with the error variance of a given model. The result is the variance that should be applied to the parameter when tests upon its significance are performed. Looking at the Variance Profiler in Figure 33, it appears that the mutation operators will require twice as strong a signal to be found significant as the rectification and random connection parameters.

The above variance results are an indicator that if all parameters were of equal importance, more test runs would likely be necessary to properly capture their effects upon the operation of the algorithm. In mixture-process design, a common strategy is to "cross" the experimental designs, meaning that either or the fractional factorial for process variables alone is replicated at each point of the mixture design or vice versa [21]. Again, in this case, a number of runs were specified, and the JMP software attempted to return the best result.

It was decided that the parameters pertaining to mutation settings were less

important than the effects of population size and the use of rectification operators, and so variance on these parameters was treated as less problematic. If useful information on these parameters was to be found, that would be useful. However, it was not considered important enough to add experimental runs. The effects of the mixture parameter levels were also left aliased. In this case, the mutation operator parameters were varied more in order to show the stability of the effects of rectification and random starts than to choose proper operator values. In any case, the desired levels of these values will depend on the details of the combinatorial space that is optimized.

The time constraints came in part from the number of trials made on each group of settings, but were mostly due to the need to debug Sindri. The Design of Experiments did at least one of its jobs very effectively, which was to put the Sindri code into almost every possible state it could be in. Every bug in indexing, miscopied chromosome strings, or other subtle error was exploited to crash the program. The uncanny ability of the DoE to crash the program, needless to say, greatly increased the time required to complete the data collection.

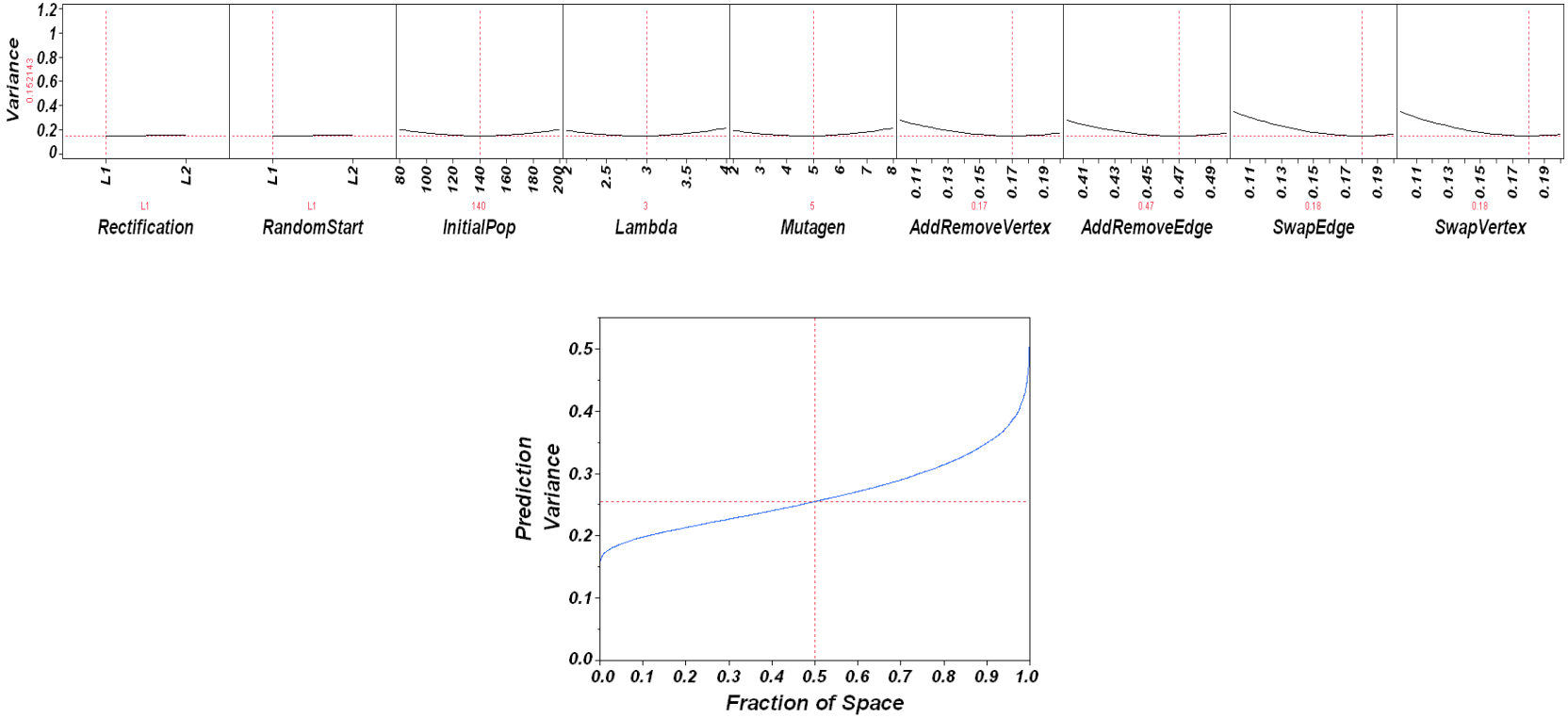


Figure 33. Design of Experiments variance information

The time to execute the genetic algorithm under a given vector of settings as listed in the DoE can be understood by considering the design space that was generated by the Excel tool. The design spaces to be tested are reproduced in Table 5 for the reader's convenience. For the purposes of this test, the stop criterion for the genetic algorithm was a fixed number of generations. Thus, the time to complete a given run from the DoE was predictable.

Table 5. Test combinatorial search spaces

	Space I	Space II	Space III
Functions Used	Lin11, Quad11, Norm11, Sig11	Lin11, Quad11, Norm11, Sig11	Lin11, Quad11, Norm11, Sig11, Quad21, Norm21, Sig21, Lin1Norm1, Quad1Sig1
Weighted?	No	Yes	No
Feasible combinations	7364	7364	142257
Number of repeats of a block allowed	2	2	1

Execution time for each experimental case in the DoE for the third test space, which had two dimensions and 142,000 viable combinations, was about six hours. This included roughly 208,000 function calls. The number of function calls was held steady by using a fixed number of generations for each case, and adjusting this number with both λ and size of initial population. A great majority of the computational time was in Pacelab's regeneration of Functional and Engineering Objects and the overhead associated with those operations.

Available Data

In order to maximize the ability to learn from testing Sindri under various conditions, some consideration was given to the type of data it would output. An HTML format was used to have the data pre-formatted for easy reading and later manipulation. The basic data, namely the fitness values of the best members of each generation, were printed on their own into a small file. More detailed data on the fitness of every member of the generation, its chromosome as represented internally within Sindri, and a reference to an image of the encoded graph, were included in a verbose file. The graph images were generated automatically by a program called NetDraw, which can take in graph descriptions as flat text files. It also has the ability to draw "nice graphs" (those easily read by people) in a batch mode, which was used to produce the images. A sample of this output is shown in Figure 34.

Auxiliary information was also embedded into the file for exploratory purposes, although it did not prove fruitful for examination. At every generation, the "distance" between two graphs was computed between the parent and child. Distance, in this case, was taken the sum of vertices and edges that had been added or subtracted to form the child from the parent. While somewhat interesting, it was hard to find a way to rigorously use this data to characterize the journey of the genetic algorithm through the search space. In addition to distance, each child that had a higher fitness value higher than its parent was highlight in orange. This was very useful to quickly spot improvements at a glance and understand where improvements in fitness came from. However, as with the distance, it was difficult to use this information in a rigorous way.

While looking through these files in the course of running the Design of Experiments, several observations can be made. First of all, the genetic algorithm is extremely adept at "cheating," and using nonsensical architectures to improve its scores. A particularly telling example was found while working with Space III. If the rectification operator was not being used, the genetic algorithm began to employ the

Results for generation 2

Best score of generation = 3.09898895148324, earned by entity 8

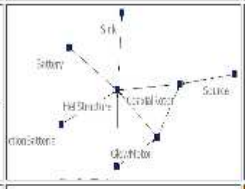
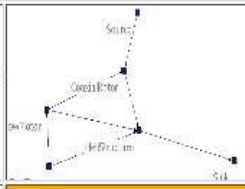
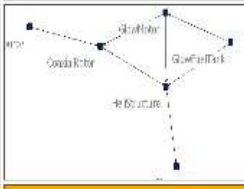
Node ID	0	1	2
Parent	0	1	2
Distance from Parent	13	11	7
Image of Architecture			
Fitness Value	0	2.74183848982823	2.74183848982823
Coded Chromosome	6 5 2 4 5 4 5 2 3 5 0 5 1 1 0 2 1 4 1 1 4 5 2 2 5 2 2 1 1 0 5 2 1 5 5 3 6 6 3 7 1 1 7 7 2 1 2 1 8	0 3 1 1 0 2 1 2 1 1 4 3 2 2 5 2 0 1 1 0 3 2 1 5 3 3 6 6 3 5 1 1 7 7 0 1 2 1 8	0 3 1 1 0 2 1 2 1 1 4 3 2 2 5 2 0 1 1 0 3 2 1 5 3 3 6 6 3 5 1 1 7 7 0 1 2 1 8
Vertex Types	CoaxialRotor GlowFuelTank GlowMotor HeliStructure Battery ElectricMotor MultifunctionBatterie Source Sink	CoaxialRotor GlowFuelTank GlowMotor HeliStructure Battery ElectricMotor MultifunctionBatterie Source Sink	CoaxialRotor GlowFuelTank GlowMotor HeliStructure Battery ElectricMotor MultifunctionBatterie Source Sink
Continuous Information		0.15 0.599978155353297 0.599978138920509 0.877766735543974 GenericAssembly.GlowFuelTank_2_2.designCapacity GenericAssembly.GlowFuelTank_2_2.designDraw GenericAssembly.GlowMotor_2_1.designFuelFlow GenericAssembly.CoaxialRotor_2_1.designDrotor	0.15 0.599978155353297 0.599978138920509 0.877766735543974 GenericAssembly.GlowFuelTank_2_2.designCapacity GenericAssembly.GlowFuelTank_2_2.designDraw GenericAssembly.GlowMotor_2_1.designFuelFlow GenericAssembly.CoaxialRotor_2_1.designDrotor

Figure 34. Sample of HTML Sindri output

Quad11, Lin1Quad1, and Quad21 vertices back-to-back in eight (out of nine possible) vertex arrangements. In this space, graphs with more vertices tended to have higher scores. Due to the number of in-degree and out-degrees of different types, no well-connected graphs had this combination of vertices. On closer inspection, it appeared that the genetic algorithm was producing high-scoring, but poorly connected graphs in order to stack the three vertices with quadratic functions together. The quadratic function, as employed in particular space, did not have a limit to its output and grew the fastest, so this made intuitive sense.

In addition to understanding where the optimizer provided counterintuitive results, it also helped in understanding were it had problems continuing its climb. One problem encountered during optimization is that the method for limiting the number of times a given function could be used in combination was probably ill-formed. Every time a candidate solution used the function more often than allowed, it was given a score of zero and effectively ignored by the rest of the optimization routine. Space II, which considered weights in its objective function, effectively limited itself to a small number of functions. This likely help the optimizer by smoothing the search space.

When the candidates with too many repeats of a given function were zeroed out, this tightly constrained the search space for Spaces I and III, because their optimal values involved the use of all of the vertices, and in different orders. Thus, the only way to reorder the vertices was to swap edge connections, and any time a new vertex of the wrong type was added, this would cause a generation of progress to be lost. This problem was exacerbated in Space III, because the addition of a new vertex would require multiple new connections to be formed in order to properly integrate it into the new graph. The new vertex problem was probably exacerbated by source-sink rectification, which would also cause the candidate to be zeroed out (with multiple connections to the sink, which also promoted cheating when it was allowed).

In making the above considerations, it can be seen that both verbose reporting from the Sindri code and the use of a standardized test space enabled multiple issues to be identified.

Main Effects Results

Once the Design of Experiments was executed on the three combinatorial spaces, the resultant data were collected and brought into a single spreadsheet. This spreadsheet was used to compare the result values to those of the test spaces. This transformed the fitness values of each solution into an ordinal number, which is a more appropriate

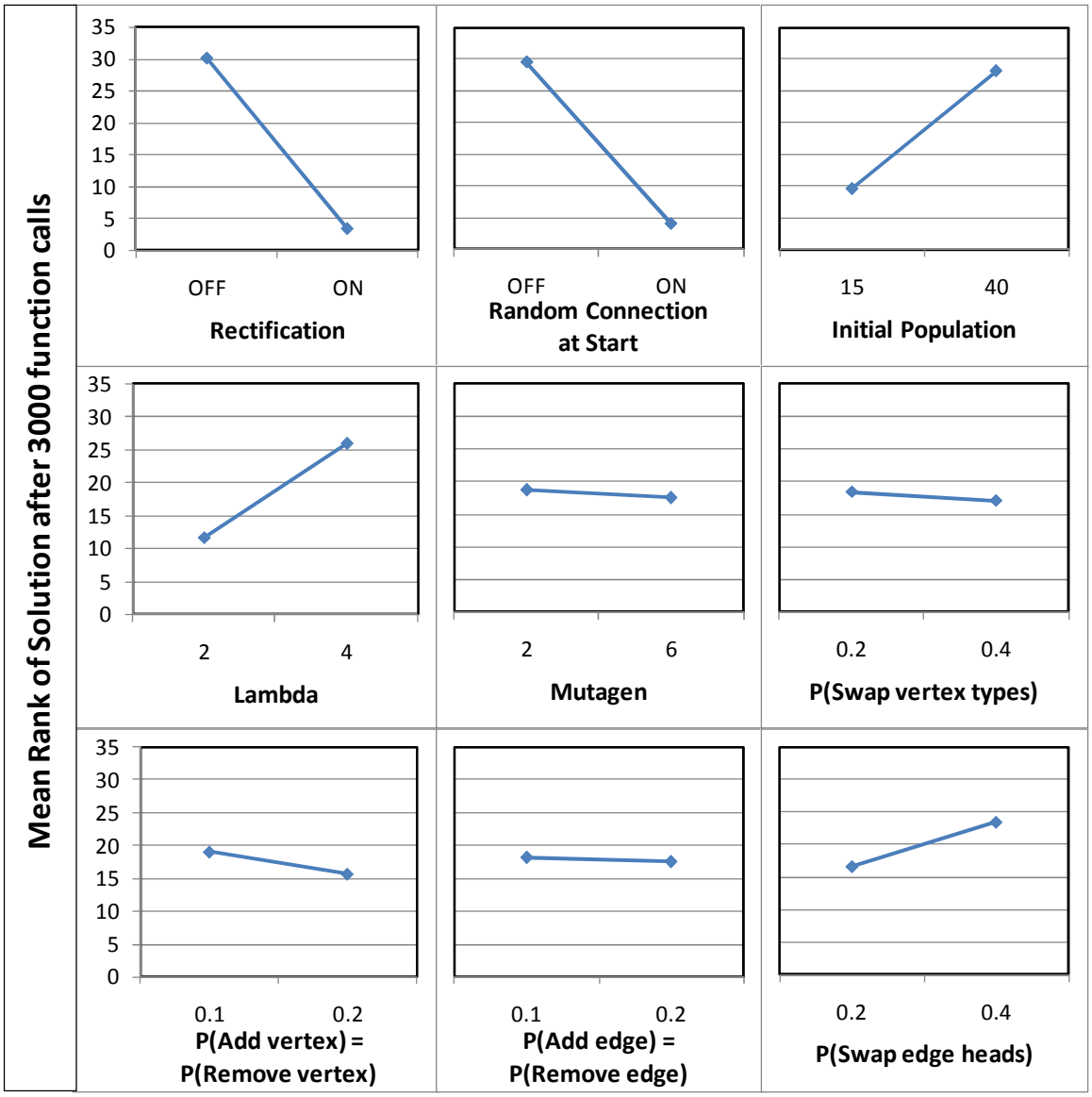


Figure 35. Main effects plot for A blocks problem (unweighted)

measure of performance of a combinatorial problem than a continuous value. The reason for this is that a large number of solutions may have a similar value, while increasing the ranked value (e.g., from 20th to 19th place) by a single number may lead to a large fitness increase.

Each vector of settings in the Design of Experiments led to a given experimental run. These runs were evaluated for the ranked value of the solution they had obtained by a given number of function calls. This became the output (Y) value to match with the

Design of Experiments for an evaluation of main effects. The tool of choice for this was a series of plots of the mean value of design factors. The results can be seen in Figure 35, Figure 36, and Figure 37. The variances are not pictured in this plots. The reason for this is that the variances of the results were greater than the level of the effect in many cases.

However, this requires some deeper consideration before dismissing all of the effects out of hand. Since the genetic algorithm is a stochastic solution strategy, there is a large amount of variation in the number of functional calls to reach a given quality of solution even with constant settings. This effect might be reduced through multiple executions of the same settings, but again, this conflicted with logistical constraints.

Another way to solve this problem is to consider the results as posted in Table 6 for the first design space. What can be noted is that in the case of good performance, such as a given input vector leading to solutions on average in the top hundred possibilities of the search space, variance is relatively small. Meanwhile, settings with mean outcomes within the top thousand possibilities often contained runs that performed well along with those that did not. A factor with this behavior implies that it tends usually toward good performance when at one setting, and a less guaranteed level of

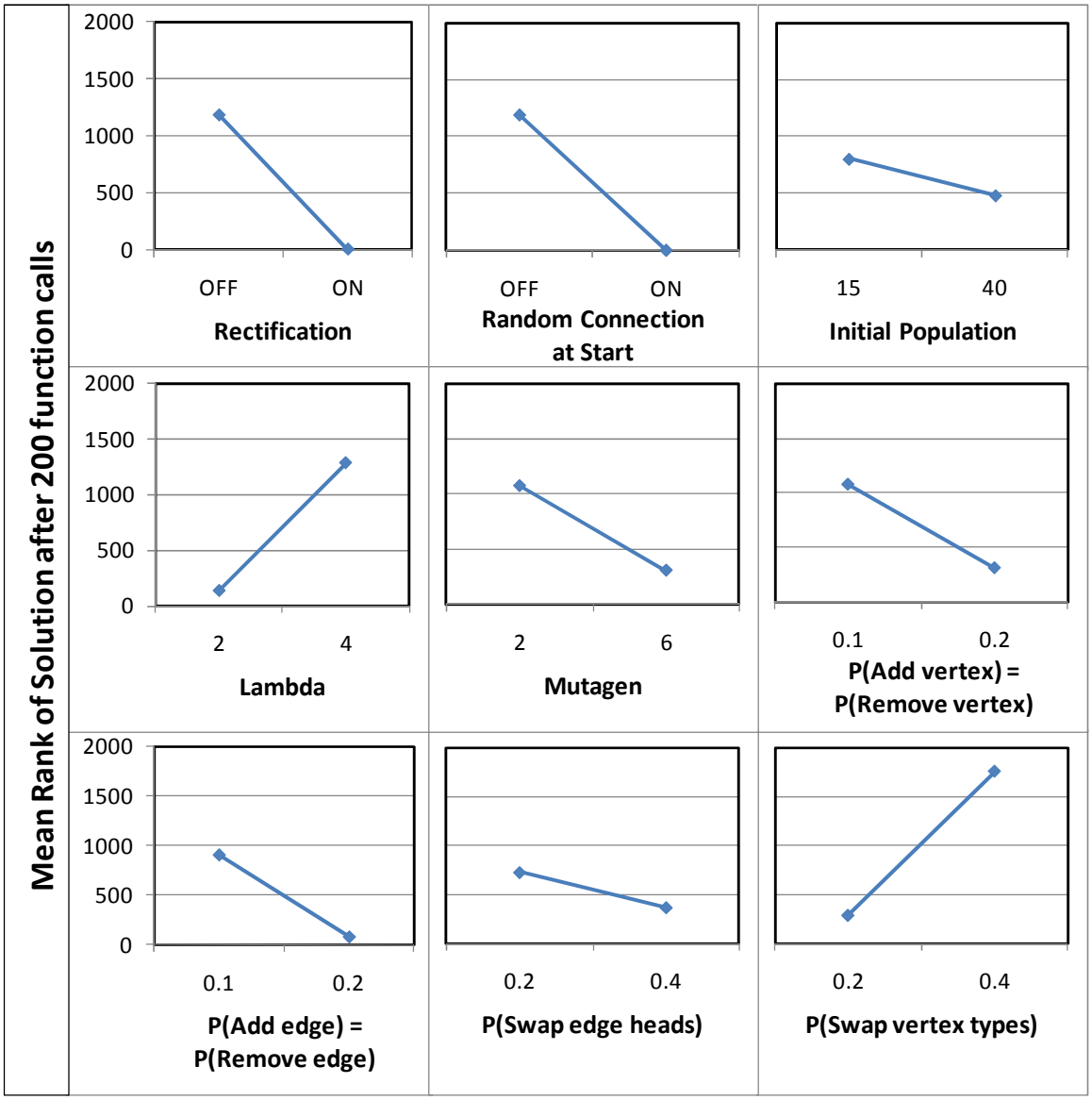


Figure 36. Main effects plot for A blocks problem (weighted)

performance when at another. Thus, there is some justification in saying that the factor has an effect on the performance of the genetic algorithm.

Looking again at Figure 35 through Figure 37, a number of trends can be identified. First of all, in all three test spaces, the source-sink rectification operator has a marked impact on performance. It is one of the strongest effects in all of these cases. Further, it appears that for the smaller spaces, starting with a well-connected graph via the random connection operator leads to an advantage. However, this is not the case with

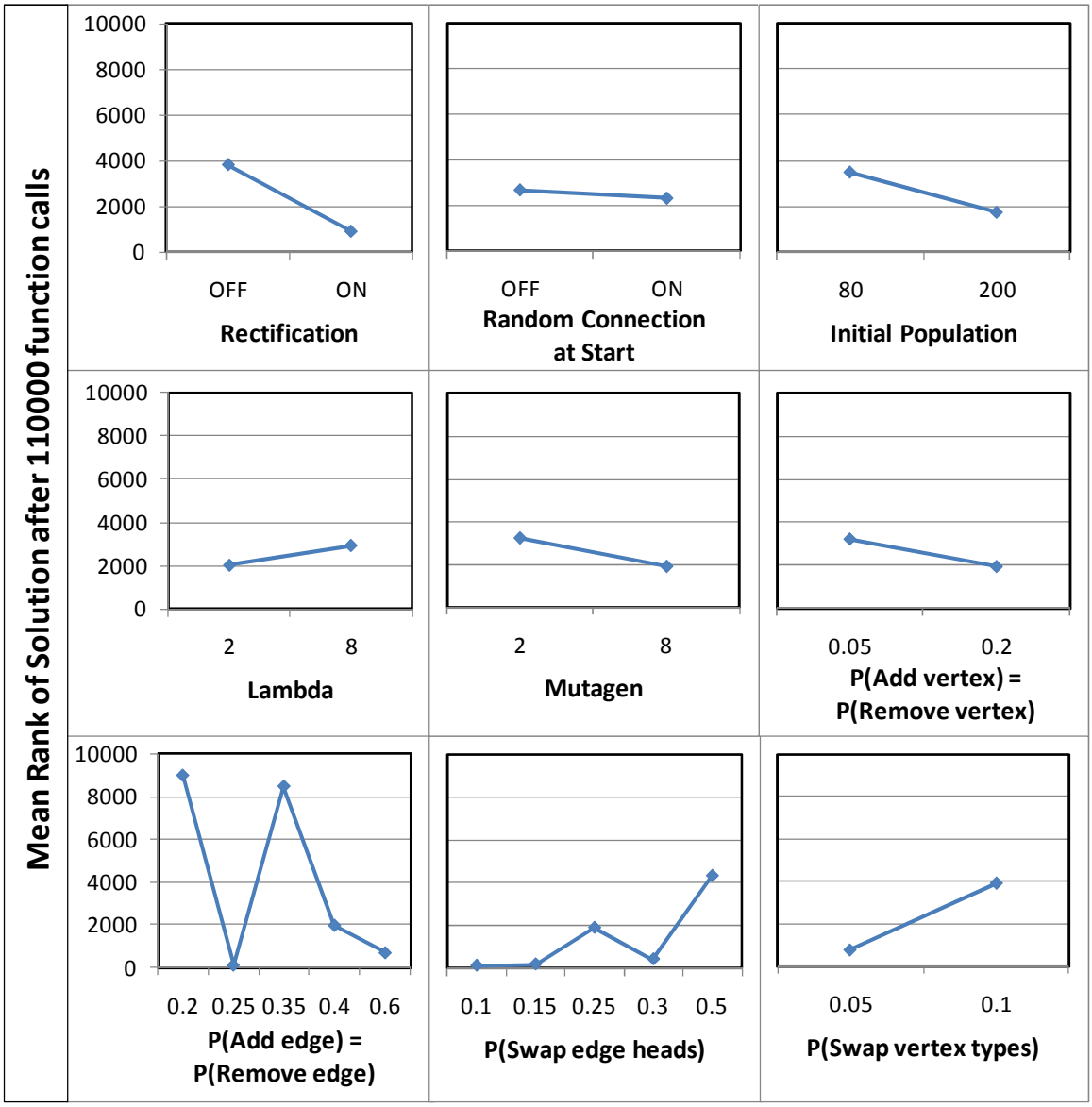


Figure 37. Main effects plot for A-B blocks problem (unweighted)

the larger design space that involves far more function calls. So, it appears that the random connection provides for a good start, but with more function calls, longer-term effects on the genetic algorithm are more important.

Table 6. Results of main effects analysis on Space I

	Mean Low	Mean High	Std. Dev. Low	Std. Dev. High
Rectification	30.09	3.33	37.33	3.08
RandomStart	29.55	4.00	37.68	4.00
InitialPop	9.73	28.22	22.48	36.76
Lambda	11.64	25.89	24.91	35.95
Mutagen	18.67	17.55	33.08	29.66
AddRemoveVertex	19.07	15.67	33.09	25.56
AddRemoveEdge	18.20	17.60	30.63	33.23
SwapEdge	16.69	23.50	28.51	41.67
SwapVertex	18.40	17.00	30.53	33.54
Rectification	30.09	3.33	37.33	3.08

The mutagen, initial population, and λ parameters have very interesting outcomes. In many cases, they provide a strong effect. However, this effect is in contradictory directions. When lower numbers for initial population are preferred, it implies that the extra members of the population are serving chiefly as wasted function calls. This would further imply that the majority of progress is due to a small number of lines of heritage. The parameter λ affects the number of "children" any given member of a population has, and defines how much of the algorithm's effort is spent on local search. There are a couple of different possibilities to explain a lack of effectiveness due to λ. The first is that the algorithm is proceeding smoothly toward optimality without much regard for direction until then end. The other is that crossover is responsible for more improvement than mutation. Meanwhile, watching effectiveness climb implies that progress in the search space is difficult and due to multiple attempts at local improvement. Finally, the mutagen represents the size of a search "hypersphere" at a given mutation. When a higher level of mutation is more effective, it implies that the search space is relatively

smooth and can be traversed rapidly. Lower levels imply a landscape with many nooks and crags that require that progress be measured and diligent.

Finally, in looking at the mutation operator probabilities, there are an array of interpretations. The most striking variations are in the case of the design space with two different types of connections. There are non-monotonic results. However, it should be noted that the experimental designs were heavily biased toward one particular setting out of the four possible. Thus, it is difficult to treat the results as useful or sensible. Looking at the results for the single-type search spaces does not provide much more insight. In the unweighted case, the mutation operator settings do not appear to have much effect. In the weighted case, the swap vertex operator did appear to have a strong effect. Since it and the other probabilities had to add to 1, the strong effect of the swap vertex probability probably influenced the results of the other three probabilities, leading to artifacts in their effect plots.

There is another way to look at the mutation operator effects showing little sense. While the main effects results show little support for one setting of the parameters over another, they do support the idea of using the parameters as noise variables. Since only one run of each data point in the DoE was utilized, this greatly strengthens the results for the other variables' influence because they cut through both the noise variables at multiple runs of the genetic algorithm on the non-noise settings. Thus, while the design of the experiments conducted on the genetic algorithm were not ideal, it does lead to useful information on its performance as a whole and upon the effectiveness of rectification and random completion operators in particular.

After considering all three search spaces together, the only clear signal that emerges is that having a well-connected starting point and using the source-sink rectification operator provide improvements in algorithm efficiency. The implications of this for the hypotheses will be fully discussed at the end of this chapter. In a nutshell, the results here show some support for Hypothesis #1 (GA's are a relatively efficient strategy

for this problem), disprove Hypothesis #2 (using purely random operators and selection to establish useful architectures does not degrade performance), and show some support for Hypothesis #3 (randomly connecting start graphs improves performance).

Some further test runs on the third test space were conducted in order to see if a new order of genetic algorithm operators would improve performance. The new test runs ran the Sindri code with an initial population of 100, λ of 2, and mutagen level of 6. The order of operators is the same as described in the Brokk method flowchart: mutation, random connection, crossover, rectification, anti-rectification, evaluation, and selection. The test runs were performed with a variety of settings for the mutation operator probabilities in a mixture design, given in Table 7.

Table 7. DoE settings for extra test runs on Space III

	AddRemove Vertex	AddRemove Edge	SwapEdge	SwapVertex
1	0.35	0.1	0.3	0.25
2	0.5	0.1	0.3	0.1
3	0.2	0.1	0.45	0.25
4	0.2	0.25	0.3	0.25
5	0.35	0.1	0.45	0.1
6	0.2	0.1	0.3	0.4
7	0.2	0.25	0.45	0.1
8	0.2	0.4	0.3	0.1
9	0.2	0.1	0.6	0.1
10	0.35	0.25	0.3	0.1

As before, the results of executing Sindri were compared with the values assigned to all members of the test combinatorial space. In this case, the first and tenth design vectors resulted in reaching the best solution within 40,000 evaluations, or roughly 1/3 of the number of combinations in design space III. Means were taken for each individual setting value in order to see if there were any trends in factor settings. This was done mostly as a curiosity, but it did appear that there were trends. They are pictured in Figure 38.

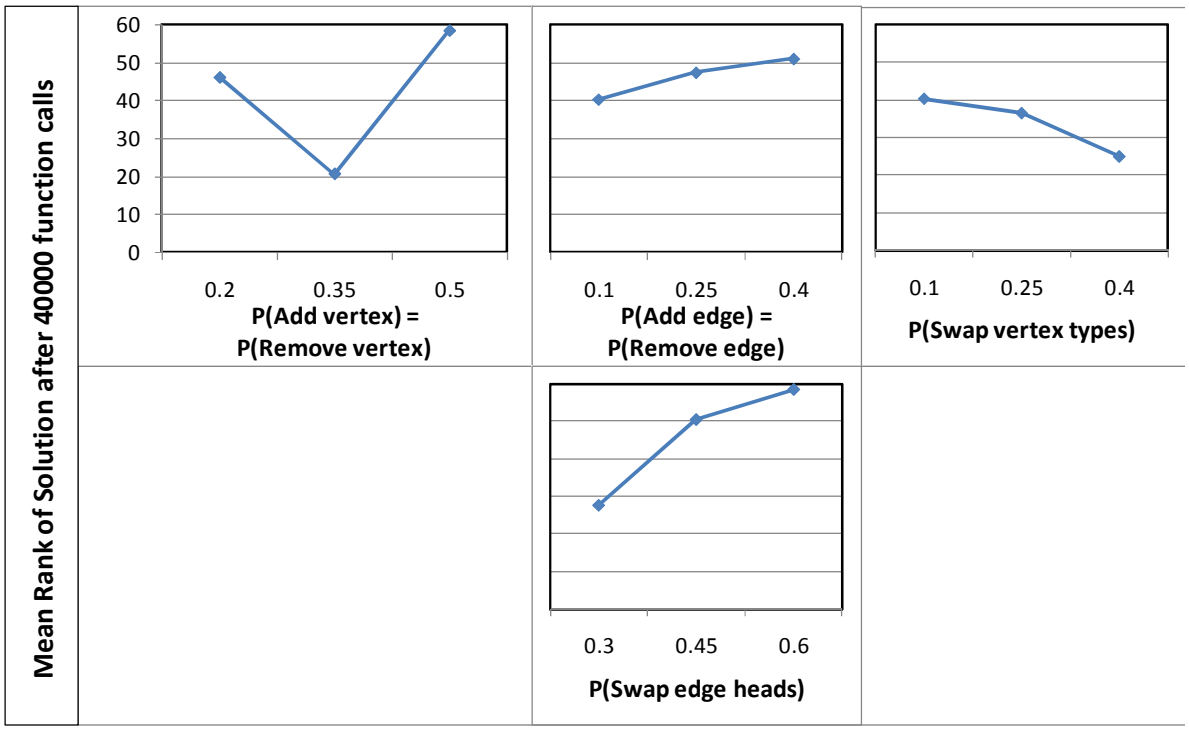


Figure 38. Main effects of added test runs on Space III

All settings above show an increase in efficiency over the previous set of Sindri operations, which went in the order of mutation, rectification, crossover, anti-rectification, evaluation, and selection. The addition of random connection after mutation served to ensure a good set of connections within the graph, and this justifies its place in the Brokk method. Further, this shows that from an arbitrary starting point, it is possible to reach optimal solutions by surveying a minority portion of the design space.

Proof of Concept Test Case

The test case brings together the technological forecasting techniques explored in the previous chapter, the genetic algorithm evaluated in this chapter, and the considerations for continuous optimization described in the previous chapter. Along with methods for visualization, it is the place that all of the previous work comes together. It also serves as a concrete example of the kinds of trades and syntheses that Sindri was designed to perform.

The test case is inspired by the Micro Autonomous Systems and Technology (MAST) program, where a number of technological components are to be brought together into new architectures. Most of this technology is beyond the near-term future, such as quantum-dot solar cells [132] or micro-ornathopters [97]. While this would serve as a good illustration of the method, it would be somewhat unanchored from previous experience. In order to retrench to something with some historical basis, the problem is couched as attaching the future of MAST-sized micro-fliers to the current and recent past trends in remote control hobby rotorcraft. While the word "hobby" typically carries the connotation of amateurish or at least less than cutting edge, in this case the hobby world has been close on the heels of the defense world.

For example, there currently is a remote control grasshopper that is based upon principles identified by the Micro Air Vehicle project. Further, this program began in the year 1996, with commercial micro helicopters being introduced in the year 2005. The main disparity between laboratory and commercial has been cost, and so there is a relatively safe assumption that the commercial micro helicopters follow similar trends to the research versions with a time lag. While there is not necessarily a perfect fit, there is the matter of finding publically available, unclassified data for the example. Again, this speaks in favor of basing the test case upon the hobby world.

The most major recent development in hobby aircraft has been the surge of electrically-powered craft into the market. Indoor fliers are miniaturized beyond what is possible with internal combustion motors. The advent of the lithium polymer battery has make it possible for an electrically-powered aircraft to have comparable performance, if not endurance, to a glow-fuel powered aircraft. Further, the commercial availability of advanced materials such as carbon fiber have allowed for their introduction into these machines. This also boosts performance levels and allows for smaller inventions to make their way to the market.

As is true with many other technologies, the adequate levels of performance for

electric fliers have opened the door to their evaluation on a metric that was rarely applied to their glow fuel predecessors. The great reduction in noise by switching to electric has allowed hobbyists to fly in areas that have been previously restricted, and, unfortunately, have caused previously permissive land owners to reevaluate their willingness to host glow fuel fliers.

With the considerations of changing metrics aside, the RC flier history serves as an excellent arena in which to test Sindri. Each step in feeding data into the tool, operating the tool, visualization, and analysis of the results will be presented here.

Step 1. Specify Problem

As mentioned in the previous section, the problem is inspired by MAST (miniaturization of existing forms and the application of multifunctional technologies) and rooted in the experience of remote control helicopters. The reason for the use of a pre-existing technology case is the ability to hold it up against the course of history. Further, the specifications of various remote control components are publically available, while actual MAST hardware is still in development.

In addition to deciding on the appropriate set of components and the general architectural problems, it is necessary to understand the desired outcome of the study. In this case, the architecture of interest is the power system for the helicopter. The goal is to capture the shift between traditional glow engine models and the electrical substitutes that have become increasingly popular in the past few years. In addition, the emergence of micro helicopters due to new electrical technologies is of interest. A further progression to structure-integrated batteries is also of interest in this particular example.

The timeframe of interest is from 1980 to 2020, with 1980 to act as a baseline year. It is also of interest to track the sensitivity of the predictions with respect to the rates of progress of three technologies: the battery, the electric motor, and the multifunctional battery. A sensitivity band of ±25% in addition to the baseline rate of

progress will be applied to the predictions. In order to understand the interactions of the rates of progress of these different technologies, a full factorial of these sensitivities for the levels of 0.75, 1, and 1.25 are desired. This leads to a design with 27 points for each year to be investigated. Ten year periods were chosen, which leads to the need to hunt for the optimum architecture in the years 1990, 2000, 2010, and 2020 subject to sensitivity factors. These altogether leads to 109 data runs if a baseline is taken in 1980.

The final aspect of the problem that requires definition is the word "optimum" for the architectures to be considered. In this case, the optimum architecture is one that can reach a given level of performance with a vehicle of minimum size. The level of performance to reach is specified as the ability to produce enough thrust to sustain hover with a roughly 10% margin for 15 minutes at full weight.

Step 2. Specify Component Models

For this example, very basic sizing relationships were used to serve as a platform for designing the components. The analysis concentrated on propulsion for the concept, and so the elements to be modeled were energy storage, power source, platform structure, and propulsor. Multiple types of storage, power source, and structure (multifunctional versus single-purpose) were considered within this analysis.

The sizing of the vehicle was performed by a simple analysis rather than a full mission examination. The main reason for this is that it was felt necessary to make any computations utilized in this demonstration reflect the current standard abilities of Sindri. In other words, the goal was to keep to a minimum the number of custom and hardcoded considerations required to make the analysis work. The problems of developing a full constraint and mission analysis for a generalized architecture are expanded upon in the future work chapter of this dissertation. The main one of these is the proper way to program the consumption of weight over time without running afoul of conditions where the vehicle may be sized to a negative weight, as happens with trajectory optimizers such

as the Program to Optimize Simulated Trajectories (POST). Thus, the analysis was designed to require only one continuous convergence loop, which was the Pacelab standard optimizer.

Before explicit models were developed, the number of components and their boundaries were considered. Seven components were crafted for the case study: a helicopter main rotor, a helicopter structure, electrical motor, battery, structural battery (combined structure and electrical loads), glow fuel engine, and glow tank. In general, the components were developed such that forces were meant to be applied to the helicopter structure in order to generate a vertical acceleration relative to the outside environment.

Modeling

The simple analysis described above was couched in the form of the requirement given when the problem was defined. The vehicle was designed to perform 15 minutes of flight at a propulsor thrust output sufficient to provide 2.5 ft / sec^2 of vertical acceleration. These quantities imposed a floor on how small the propulsor and the energy source could be. Their interactions with other elements of the architecture helped to bound the limits of those components as well.

Most of the components are empirically sized. Data on motors, engines, batteries, fuel cells, and remote control rotorcraft were collected in order to establish empirical equations for this analysis. Their sources are discussed in the next section.

The propulsor analysis is a very simple sizing relationship with empirical factors to provide a connection between thrust and power. It is built from the definition of the rotor figure of merit [93]:

$$M = \frac{C_{P_{ideal}}}{C_{P_{input}}} = \frac{{C_T}^{3/2}/\sqrt{2}}{C_{P_{input}}}$$

which can have the coefficients of power, C_P and thrust, C_T expanded and the

result rearranged to become:

$$T = \left(M \cdot P_{in}\sqrt{2\rho A}\right)^{2/3}$$

where ρ represents the air density, T is thrust, P is power, and A is the area swept by the rotor.

The helicopter structure provides the total weight of the vehicle. This weight is calculated on an empirical basis from other remote control helicopters ranging in size from fliers with six inch main rotors to those with five foot rotors. The total weight of the helicopter was regressed against both the size of the motor (as a dominant determinant of structure size) and the weight of all of the components within the structure (as a determinant of the loads to transfer and somewhat the structural size). These two regressions were taken separately, and the average of their results is given as the total weight of the helicopter. The weights are computed in pounds and the sizes in feet. The forms of these equations are:

$$W_1 = 3.902 \cdot \sum W_{load} - 0.033$$

$$W_2 = 0.205 \cdot \left(\sum d_{motor}\right)^3 + 0.01 \cdot \sum d_{motor} + 0.025$$

$$W_{structure} = \frac{W_1 + W_2}{2}$$

The helicopter structure is also assigned the task of reporting the effectiveness of the thrust produced by the propulsor. This effectively means that both a propulsor and a structure are required to be included in a candidate architecture in order for it to be given a non-zero value in evaluation.

The next component model to describe is that of the glow motor. Like the structure, it was empirically modeled from a set of hobby motor data. In this case, the independent variable was deemed to be design fuel flow. This was regressed to obtain the engines' maximum dimensions, weights, and output powers. In this case, weights are in pounds, fuel flow is in pounds per hour, lengths are in inches, and power is in watts.

The regressed equations for the glow engine are:

$$d_{engine} = -0.325 \cdot \dot{W}_{fuel}{}^{2} + 1.52 \cdot \dot{W}_{fuel} + 1.54$$

$$W_{engine} = 0.468 \cdot \dot{W}_{fuel} - 0.0125$$

$$P_{engine} = 544.23 \cdot \dot{W}_{fuel}{}^{1.182}$$

The final component to discuss in this section is the fuel tank. The fuel tank calculation basically converts a capacity requirement into a maximum dimension for a cylindrical tank.

The rest of the components are also modeled empirically, but their models fit into the category of extrapolative technology models, and so they will be discussed there. Although all of the components in this case have been modeled mostly on empirical relationships, there is nothing that prevents analytic models from having technology factors. For example, it would be conceivable that the electric motor would have a theoretical model describing its power output on a basis of permanent magnet strength, and voltage and current run through the armature. The magnet strength might then be a technological factor that is modeled with respect to time.

Data Collection

The helicopter structure, batteries, glow motors, and electric engines are all currently commercially available. Thus, the specifications of current-day equipment were taken from hobby shop vendors and manufacturers and compiled into datasets. The resultant information can be seen in Appendix D. These data were taken together and transformed into empirical relationships to predict the weight of the helicopter structure and the performance of the glow fuel internal combustion engines. The glow engines were considered to be technologically mature, and so not in need of a prediction for future for performance.

Predicting the performance of the batteries and motor over time required the next set of data to be collected. Research on the history of electric motors revealed that two

major trends in direct current (DC) motors were increasing specific power due to advances in materials and increasing efficiency. The advances in materials were primarily in lightweight, high-strength permanent magnets. The history of these advances [52], [127] were used to model the change in expected weight of a motor with a given power output.

The efficiency increase was especially pronounced in the past few years in remote control aircraft with the availability of inexpensive brushless motors to replace brushed motors. This was achieved by making commutation a product of digital controls rather than mechanisms inside the motor. Common efficiencies for brushless and brushed motors, along with an estimated time of introduction were combined to provide a curve for efficiency advances.

The performance of batteries over time was relatively simple to gather background data on. Multiple sources provide information of average specific energy and specific power for various cell chemistries. Taking this information together with the date of introduction for various chemistries allowed for the development of multidimensional S-curves, to account for the fact that some chemistries have better specific power while others have better specific energy.

The final technology to understand was the structural battery [142], currently under investigation at the Army Research Laboratory (ARL). This is a battery based specifically on the lithium polymer battery, since this battery type tends to use carbon compounds as part of the anode. The bulk separator material has also been considered for structural strengthening. The ideal version of this battery would have the structural strength and stiffness of a pure carbon composite while also possessing the full specific power and specific energy of the lithium polymer battery.

In order to develop a future performance for this battery, multiple liberties were taken from existing data. The existing literature indicates that current structural battery tests are performing at about 10 percent of the ideal electrical performance and 30 percent

of the structural performance. A final product was envisioned by the author as having 60 percent electrical performance and 90 percent structural performance. Further, the inflection point for technological development was set to be the year 2020. Thus, the future predictions developed from this information are meant only to be illustrative of the types of results the technique can provide.

Sindri Setup / Tagging

The tagging of each component describes how the various architectures possible by component combinations will be assembled. Thus, it is important to spend some consideration on it. The propulsor is an easy start, in that it transforms power into thrust, and so that is a straightforward tag. Next are the storage components, namely the glow fuel tank and the batteries. They are given fuel weight flow and electrical flow as outputs. They are also given a specialized input, "tank," that was used to indicate that the quantity of matter or energy within these components are the quantities that determine the endurance of the vehicle. The motor and engine are tagged in a manner appropriate to their particular transformation of fuel or electrical power into shaft power.

The helicopter structure required a great deal more consideration. It is the stand-in for the integrated helicopter, and is empirically modeled. This means that it will be similar in on-board components and shape to current remote control helicopter structures. The empirical model was developed to account for both the size and the weight of the main engine, along with the weight of all components placed on the structure. Several inputs were set aside for the weights and sizes, so that every engine and component loaded into the architecture would be provided a space for connection. This also ensures that the detrimental effects of redundant components will be properly accounted. The final input is a thrust input, to be received from the propulsor. The only output of the structure is an acceleration, which marks the movement of the vehicle through the

environment, and provides Sindri a handle to require at least 2.5 ft/s^2 to size the main rotor.

Summary

In order to make clear the results of this step, a summary table is provided in Table 8. This table highlights both the relative size of the combinatorial design space compared to the earlier test and the level of fidelity of modeling. The design variables affect the performance calculations of the configuration and can be selected by the user. The inputs, however, are driven by connections between components, and so are not user-selected, but rather the result of outputs from other components. Since design variables affect outputs, they do have effect on the architecture as a whole. Also, in the case of input power or fuel, it should be noted that these are maximum values, which size the engine, but a "percentage" variable is also available to alter the actual input.

Table 8. Test case component properties

Component Name	Design Variables	Input tags	Output tags	Component Model Notes
HeliStructure	None	weight Load (x5), motor Size (x4), thrust	accel-eration	Empirical sizing based on RC helicopters, scales with engine size and total subsystems weight
Rotor	rotor diameter	power, air density	thrust	Simple figure of merit calculation to relate shaft power and thrust
Motor	input power	energy Dot	power, weight Load, motor Size	Empirical data for weight and size, simple efficiency factor between input and shaft power
Battery	capacity, maximum power, specific energy (tech. trade)	tank	energy Dot, weight Load	Uses specific energy and historic trends to determine energy density, specific energy and energy density and specific power give battery sizes
GlowEngine	input fuel flow	weight Dot	power, weight Load, motor Size	Relates size and weight to fuel flow by empirical data
GlowTank	capacity, maximum draw, fineness ratio	tank	weight Dot, weight Load	Calculates volume and weight from fuel weight capacity, also generates dimensions using fineness ratio and cylindrical form
Multifunction Battery	capacity, maximum power, specific energy (tech. trade)	tank	energy Dot, weight Load	Sizes like the motor, but with an additional variable of structural strength, reduces its own weight based upon percentage own strength against pure structural member

Step 3. Identify Technology Factors

The three components that depend on a knowledge of technology advance for their computation are the electrical battery, the electric motor, and the multifunctional battery. The process of data collection for these components has already been discussed. During that process of data collection, a series of technology factors were identified as important and thus required modeling via either single-dimensional or multi-dimensional S curves.

Battery chemistries typically have three major performance metrics: specific energy (energy per mass), energy density (energy per volume), and specific power (power per mass). Of these, specific energy and specific power were chosen as relevant parameters, since weight would size the main rotor. These two quantities were also chosen due to the fact that they would set up a tension between endurance and thrust constraints.

The quantities of interest to the electric motor were in its ability to produce power efficiently and at the smallest possible size and weight. Originally, the goal was to obtain these quantities directly through multiple generations of motors, but regular data were not available for old motors. Thus, proxy technological factors were identified for the motors based on their construction. These proxy factors were efficiency (as determined by brushless versus brushed performance) and permanent magnet strengths, which provide a proxy for the decrease in motor weight over time.

Finally, the multidimensional battery was considered. Unlike the other two components, it is still at the prototype stage of development. Thus, only data for its beginning state is available. As described in the data collection discussion of the last section, the factors of interest are the structural strength and the specific energy of the battery. Since the battery is based on a single chemistry, it is assumed that specific power tracks with improvements in specific energy.

Step 4. Fit Curves to Technology Data

This step was performed by taking the data that were gathered in collected in Appendix D and applying the MATLAB code listed in Appendix C to it. Single-dimensional S-curves were also created where appropriate.

The multidimensional S-curve for the battery, which represents multiple generations of cell chemistries is:

$$t = 2034.35 - 5.65 \cdot X_{spec.energy} - 8.1 \cdot X_{spec.power}$$

$$X_{spec.energy} = ln\left(\frac{1500 - E/W_{current}}{E/W_{current} - 35}\right), X_{spec.power} = ln\left(\frac{5000 - P/W_{current}}{P/W_{current} - 100}\right)$$

The following equation is imprecise, but for this model, energy density was taken to be a function of specific energy:

$$E/V = 13.47 \cdot exp(0.00816 \cdot E/W)$$

As mentioned before, the electric motor's technology curves were developed by way of proxies in power efficiency and permanent magnet strengths. In developing the progress in motor weight due to permanent magnet strengths, it was assumed that current-day magnets account for roughly 33% of the motor's weight. This results in the following equations:

$$W_{motor} = \eta_{magnet} \cdot 0.33 \cdot \left(-2.829 \cdot 10^{-8} \cdot {P_{in}}^2 + 1.421 \cdot 10^{-4} \cdot P_{in} + 0.00388\right)$$

$$\eta_{magnet} = \frac{62.5}{\frac{108.5}{1 + exp\left(-\frac{t - 1996}{16}\right)} + 1.5}$$

$$d_{motor} = \eta_{magnet} \cdot 0.2 \cdot \left(0.0261 \cdot 10^{-2} \cdot {P_{in}}^{0.282}\right)$$

$$\frac{P_{out}}{P_{in}} = \frac{0.45}{1 + exp\left(-\frac{t - 1995}{6}\right)} + 0.5$$

The final component to be examined is the multifunctional battery. The original plan for this was to develop a version of the helicopter structure component that output acceleration (from the force applied to it) as well as a power output to the electric motor.

This is possible within Sindri in general, but the specific arrangement of components lead to a cyclic graph, which Sindri cannot work with currently. Thus, a workaround was to make the multifunctional battery like a normal battery, except that its weight decreases for a given energy storage as time progresses. The multidimensional S-curve between specific energy and structural strength (as a fraction of the strength of the composites forming the helicopter structure) is below:

$$t = 2019.2 - 3.22 \cdot X_{spec.energy} - 4.58 \cdot X_{strength}$$

$$X_{spec.energy} = ln\left(\frac{122.3 - E/W_{current}}{E/W_{current} - 2.23}\right), X_{strength} = ln\left(\frac{103.7 - \eta_{current}}{\eta_{current} - 4.33}\right)$$

In the above equations, strength is a percentage and specific energy is in watt-hours per kilogram.

Once all of the S-curves were fit to technology development data, the various component models were rendered complete. All of the information developed up to this point was loaded into the appropriate Pacelab projects and prepared for the execution of the Sindri code.

Step 5. Execute Sindri code and Step 6. Build Results Database

Once all of the models were developed, they were rendered into C# code and loaded into the appropriate Engineering Objects in the Pacelab Knowledge Designer. Design variables were identified and given the "design(Parameter)," "design(Parameter)Min," "design(Parameter)Max" pattern. Also, multidimensional S-curves have one degree of freedom once the time has been specified. Thus, there was also a need to develop a "techMetric1," "techMetric1Min," and "techMetric1Max" pattern. Both this and the design patterns are necessary to allow Sindri to develop side constraints for the continuous optimization problem sets it structures.

Table 9 shows a vector of the years and sensitivities that defined the data runs executed by Sindri. The results are also given within that data table. At each data point,

the Sindri code was executed to find the optimum solution. The optimal solution was defined by the following objective function:

$$Z = \frac{a_{structure}}{\sqrt{\sum_i d_i}}$$

where a represents the acceleration of the helicopter structure and d_i is the maximum dimension of the i-th component in the architecture. Sindri was instructed to target an acceleration of 2.5 ft/s^2, which was achieved by a Pacelab-native capability to reverse the roles of input and output variables. The roles of the "goal" parameter of the sink vertex (originally dependent) and the diameter of the main rotor (originally independent) were the two that were switched.

Table 9. Results database for test case

Year	$K_{battery}$	K_{motor}	K_{multi}	Dominant Architecture	Drotor (ft)
1980	1	1	1	GlowMotor	0.768
1990	1	0.75	0.75	GlowMotor	0.768
1990	0.75	0.75	1.25	GlowMotor	0.768
1990	1.25	1	1.25	GlowMotor	0.768
1990	0.75	1.25	1	GlowMotor	0.768
1990	1	1.25	1	GlowMotor	0.768
1990	1.25	1.25	0.75	GlowMotor	0.768
1990	1	1.25	1.25	GlowMotor	0.768
1990	1	0.75	1.25	GlowMotor	0.768
1990	1	1	1.25	GlowMotor	0.768
1990	1.25	1.25	1	GlowMotor	0.768
1990	1.25	1.25	1.25	GlowMotor	0.768
1990	1.25	0.75	1.25	GlowMotor	0.768
1990	0.75	1	0.75	GlowMotor	0.768
1990	0.75	0.75	0.75	GlowMotor	0.768
1990	1.25	1	0.75	GlowMotor	0.768
1990	0.75	1.25	1.25	GlowMotor	0.768
1990	1	1.25	0.75	GlowMotor	0.768
1990	1.25	0.75	1	GlowMotor	0.768
1990	0.75	1.25	0.75	GlowMotor	0.768
1990	0.75	1	1	GlowMotor	0.768
1990	1	1	1	GlowMotor	0.768
1990	0.75	1	1.25	GlowMotor	0.768
1990	0.75	0.75	1	GlowMotor	0.768
1990	1.25	0.75	0.75	GlowMotor	0.768
1990	1	0.75	1	GlowMotor	0.768

1990	1	1	0.75	GlowMotor	0.768
1990	1.25	1	1	GlowMotor	0.768
2000	1	0.75	0.75	Electric	0.693
2000	0.75	0.75	1.25	GlowMotor	0.768
2000	1.25	1	1.25	Electric	0.587
2000	0.75	1.25	1	GlowMotor	0.768
2000	1	1.25	1	Electric	0.707
2000	1.25	1.25	0.75	Electric	0.534
2000	1	1.25	1.25	Electric	0.707
2000	1	0.75	1.25	Electric	0.693
2000	1	1	1.25	Electric	0.648
2000	1.25	1.25	1	Electric	0.534
2000	1.25	1.25	1.25	Electric	0.534
2000	1.25	0.75	1.25	Electric	0.515
2000	0.75	1	0.75	GlowMotor	0.768
2000	0.75	0.75	0.75	GlowMotor	0.768
2000	1.25	1	0.75	Electric	0.587
2000	0.75	1.25	1.25	GlowMotor	0.768
2000	1	1.25	0.75	Electric	0.707
2000	1.25	0.75	1	Electric	0.515
2000	0.75	1.25	0.75	GlowMotor	0.768
2000	0.75	1	1	GlowMotor	0.768
2000	1	1	1	Electric	0.648
2000	0.75	1	1.25	GlowMotor	0.768
2000	0.75	0.75	1	GlowMotor	0.768
2000	1.25	0.75	0.75	Electric	0.515
2000	1	0.75	1	Electric	0.693
2000	1	1	0.75	Electric	0.648
2000	1.25	1	1	Electric	0.587
2010	1	0.75	0.75	Electric	0.314
2010	0.75	0.75	1.25	Electric	0.488
2010	1.25	1	1.25	Electric	0.190
2010	0.75	1.25	1	Electric	0.423
2010	1	1.25	1	Electric	0.268
2010	1.25	1.25	0.75	Electric	0.181
2010	1	1.25	1.25	Electric	0.268
2010	1	0.75	1.25	Electric	0.314
2010	1	1	1.25	Electric	0.281
2010	1.25	1.25	1	Electric	0.181
2010	1.25	1.25	1.25	Electric	0.181
2010	1.25	0.75	1.25	Electric	0.214
2010	0.75	1	0.75	Electric	0.457
2010	0.75	0.75	0.75	Electric	0.488

2010	1.25	1	0.75	Electric	0.190
2010	0.75	1.25	1.25	Electric	0.423
2010	1	1.25	0.75	Electric	0.268
2010	1.25	0.75	1	Electric	0.214
2010	0.75	1.25	0.75	Electric	0.423
2010	0.75	1	1	Electric	0.457
2010	1	1	1	Electric	0.281
2010	0.75	1	1.25	Electric	0.457
2010	0.75	0.75	1	Electric	0.488
2010	1.25	0.75	0.75	Electric	0.214
2010	1	0.75	1	Electric	0.314
2010	1	1	0.75	Electric	0.281
2010	1.25	1	1	Electric	0.190
2020	1	0.75	0.75	Electric	0.171
2020	0.75	0.75	1.25	MultiFunctional	0.089
2020	1.25	1	1.25	MultiFunctional	0.085
2020	0.75	1.25	1	Electric	0.261
2020	1	1.25	1	Electric	0.156
2020	1.25	1.25	0.75	Electric	0.080
2020	1	1.25	1.25	MultiFunctional	0.083
2020	1	0.75	1.25	MultiFunctional	0.083
2020	1	1	1.25	MultiFunctional	0.083
2020	1.25	1.25	1	Electric	0.080
2020	1.25	1.25	1.25	Electric	0.080
2020	1.25	0.75	1.25	Electric	0.119
2020	0.75	1	0.75	Electric	0.266
2020	0.75	0.75	0.75	Electric	0.282
2020	1.25	1	0.75	Electric	0.110
2020	0.75	1.25	1.25	MultiFunctional	0.085
2020	1	1.25	0.75	Electric	0.156
2020	1.25	0.75	1	Electric	0.119
2020	0.75	1.25	0.75	Electric	0.261
2020	0.75	1	1	Electric	0.266
2020	1	1	1	Electric	0.159
2020	0.75	1	1.25	MultiFunctional	0.083
2020	0.75	0.75	1	Electric	0.282
2020	1.25	0.75	0.75	Electric	0.119
2020	1	0.75	1	Electric	0.171
2020	1	1	0.75	Electric	0.159
2020	1.25	1	1	Electric	0.110

The “K” factors, $K_{battery}$, K_{motor}, and K_{multi}, require some explanation. Each of the

components in the subscripts is a component with a technology growth curve in the model. In each of these models, time is a driving variable for the technology parameters. The "K" values modify the time to be used within the models. This is performed by the following transformation:

$$t_{effective} = t_{current} \cdot K_{component} + t_{base}$$

The effective time is then applied to the appropriate component in order to accelerate or decelerate its development. In this test case, t_{base} is set to be the first time investigated, 1980. Thus, a battery in the year 2000 with a K value of 0.75 has an effective time of 1995. When the Sindri code is set to the year 2000 then, the trade between the two technology characteristics, specific energy and specific power, is enacted as if the year was set to 1995.

The above formulation relates to an important idea about the nature of S-curves. Typically, progress along the vertical axis of the S-curve is governed by physical law. The entry point is the performance of some crude version of the device, often just developed enough to show its function for the first time. The limit is certainly due to the laws of physics, which govern the maximum performance a given physical effect can generate based on a certain level of inputs. The horizontal axis of the S-curve, by contrast, is driven much more by programmatic factors. The various levels of effort put into a research project (up to a point, of course, as illustrated by such books as the Mythical Man-Month) determine the rate of its progress. Thus, the effective time formulation for sensitivity is designed to capture this particular effect, which models uncertainty in either a competitor's level of commitment to a given project, or the rate of progress due to a given level of effort. The notional speeds of a standard and a crash program are depicted in Figure 39.

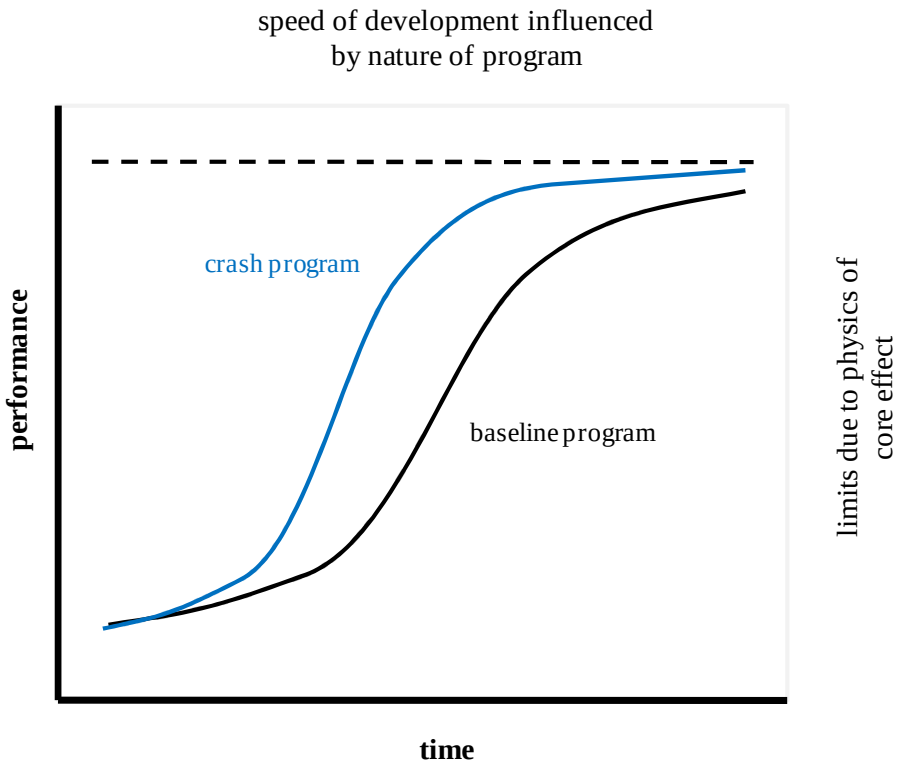


Figure 39. Illustration of factors shaping technology S-curve

In the database above, the dominant architecture column refers to one of three named architecture types. Each of these architectures have four components. The "GlowMotor" architecture is composed of the glow engine, helicopter structure, rotor, and glow fuel tank. The "Electric" architecture combines the electric motor, helicopter structure, rotor, and battery. Finally, the "MultiFunctional" architecture is the same as the "Electric," but replacing the battery with the multifunctional battery.

The three architecture options discussed above are not the only architectures that were developed by the genetic algorithm. However, all of the other options were deficient, either because they were not complete (the acceleration value from the structure was negative one gee) or because they had extra components, such as extra engines, that added weight but no performance to the vehicle. The extra components added to the required thrust, and thus led to solutions that were unable to compete against the solutions that were more economical in their selection of components.

With a database of results in hand, it is time to progress to the next and final step. The results are visualized in the next section.

Step 7. Visualize Results

Before deciding upon the appropriate visualization, it may be useful to recall the characteristics of the test case scenario. The timeframe for simulation of the test case is from the years 1980 to 2020. Sensitivity information for this test case is generated by applying a multiplier to the current year for each of three technology components. In this case, the sensitivity parameters are set to 0.75, 1, and 1.25. These values are applied to each of the technology components through a full factorial design. The design is replicated for the years 1990, 2000, 2010, and 2020. The response values of interest were the diameter of the main rotor, a proxy for vehicle size, and the type of architecture that was dominant.

Even with a small number of components and sensitivities, the data are sufficiently rich to merit multiple views and a disciplined walkthrough. This walkthrough should also serve as an indicator of the types of trends and results that should be expected from a scaled-up version of this process. The goal of this will be to illustrate how the database developed by using the Brokk method can be used to understand how the dominant architecture may change over time. Seeing both the baseline scenario and scenarios changed by sensitivity adjustments may provide insight into opportunities in the future. This in turn can provide strategic insight about which architectures to examine for further potential.

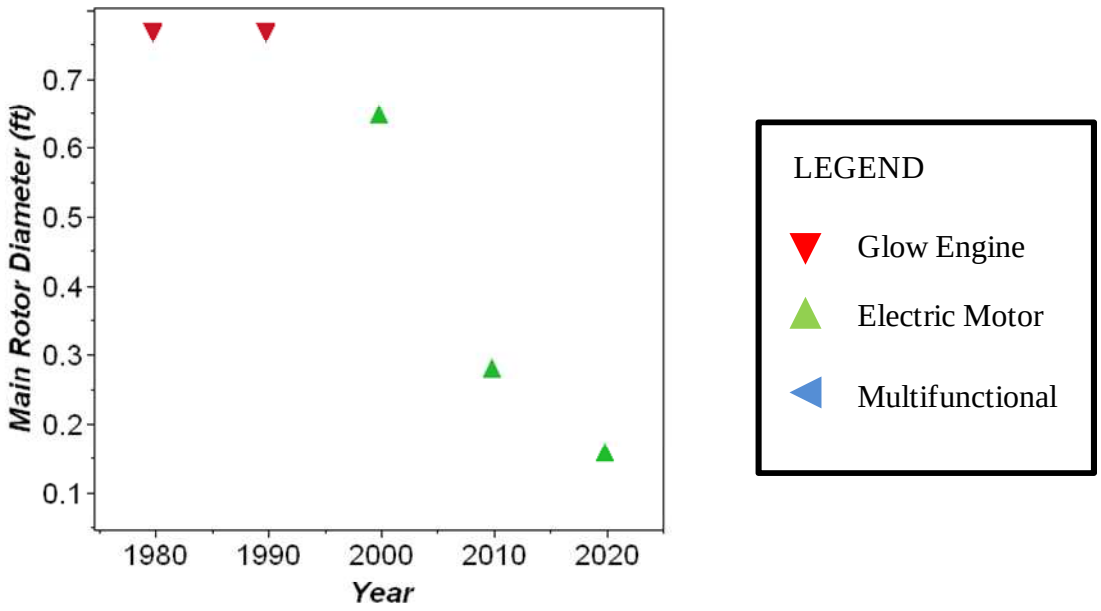


Figure 40. Main rotor diameter as function of time, baseline forecast

Although the cost function used in the execution of Sindri was a norm of all component sizes, a metric of interest is the rotor diameter of the helicopter. The progression of this value can be used as a proxy for the decreasing size of hobby helicopters allowed by improvements of electrical technology. The motor rotor diameter progression over time can be seen in greater detail in Figure 40. In this baseline case, a major jump in rotor size can be seen between 2000 and 2010. This is consistent with the introduction of micro helicopters in. The rotor diameter for the Pico Z [29], introduced in 2006, for example, is 5.25 inches, or 0.44 feet. The helicopter also has about 10 minutes worth of endurance on its battery. It also appears that the first group of commercial palm-sized helicopters were introduced in the year 2005. The main developments that are involved in the micro helicopter are the brushless DC motor, which are highly efficient and have fewer parts than brushed motors, and lithium polymer batteries.

The prediction made for the size of the helicopters is a little bit optimistic, but this may be explained by the low fidelity of the model, and is highly sensitive to the figure of

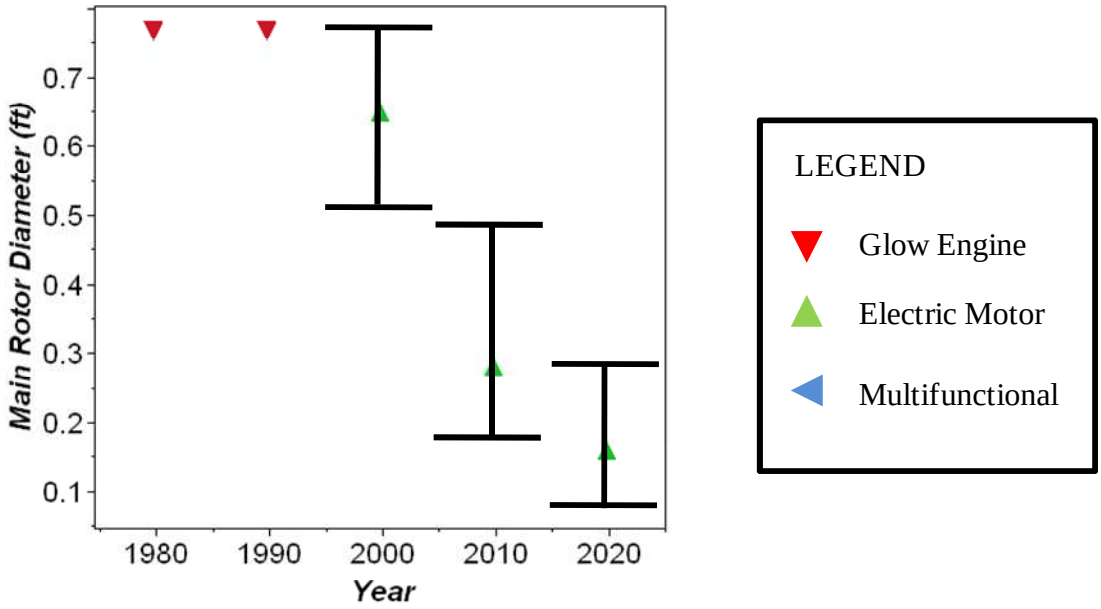


Figure 41. Main rotor diameter as function of time, baseline forecast with sensitivities

merit that was input for the propulsor component model. The figure of merit was chosen to be 0.4, which is conservative relative to a high-performance airfoil for low Reynolds numbers as indicated in [47]. The commercial micro helicopter airfoil may be much lower performing.

The next figure of interest is Figure 41, where the results of sensitivity have been overlaid. The error bars show the full dispersion of results that are due to the variation of all component K values from 0.75 to 1.25. In the major year of change, 2010, the results are very sensitive to change. Only a ± 25% shift in the rate of progress from 1980 results in a ± 100% change in expected main rotor diameter. This corresponds to a shift in the introduction of a given technology level by ± 7.5 years. In a competitive situation, this would make the year 2010 important to watch.

The sensitivity of dominant architecture to the K parameters can be seen by generating two-dimensional plots that show the year on one axis and one of the K parameters on the other. Since there are three such parameters, there are three such plots,

which can be seen in Figure 42, Figure 43, and Figure 44. In each of these figures, the value of the K parameter is set to be 1 (although this can be changed to other settings if desired). This means that each of the plot shows a sensitivity in just one dimension at a time. Sluggish development in electrical technology causes the glow motor to remain dominant up to the year 2000, and rapid development of multifunctional batteries causes its introduction in the year 2020.

While all of the forecast information, including sensitivities, can be contained within these two-dimensional plots, it can be difficult to get an intuitive sense of all of it at once. In order to do so, it can be useful to visualize in more dimensions, in this case four. Figure 45 and Figure 46 represent a quasi-four-dimensional plot of all of the sensitivity data. The two images are of the same plot, but rotated in order to give the reader a better view and ability to perceive depth. They show data in three Cartesian dimensions, while the fourth is represented by clustering around each three-dimensional point. In each cluster of three points, the top point has a $K_{multifunctional}$ value of 1.25, the center a value of 1, and the bottom a value of 0.75. The center point of the lower-left-most cluster in Figure 46 thus has a quadruple value of (1990, 1.25, 1.25, 1) is the order of values is year, $K_{battery}$, K_{motor}, $K_{multifunctional}$.

When looking at the two plots, it can be seen that the multifunctional battery only becomes viable with an effective time of 2030 ($K_{multifunctional}$ at 1.25 in 2020). It can also be seen that the time of forecasted transition between glow powered and electrical architectures is between 1990 and 2005. By looking between the two plots, it can also be seen that, when sensitivities are considered, there is a kind of inclined plane that shows more angle in the direction of battery development rate than the motor development rate. Thus, a company developing the helicopters in 1980 would be advised to start investigating electrical competencies and tracking the progress of electrical components more closely. The focus may be further placed upon energy storage technology due to the sensitivity information.

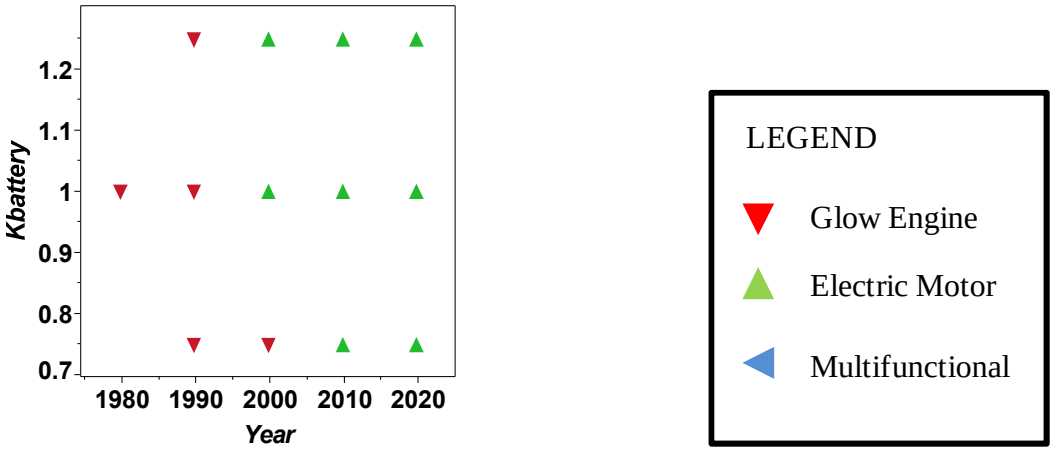


Figure 42. Sensitivity plot for $K_{battery}$, $K_{motor} = K_{multifunctional} = 1$

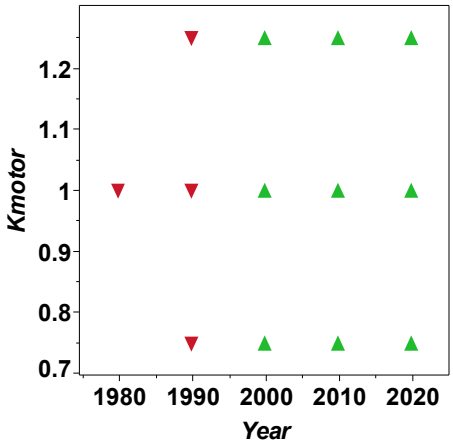


Figure 43. Sensitivity plot for K_{motor}, $K_{battery} = K_{multifunctional} = 1$

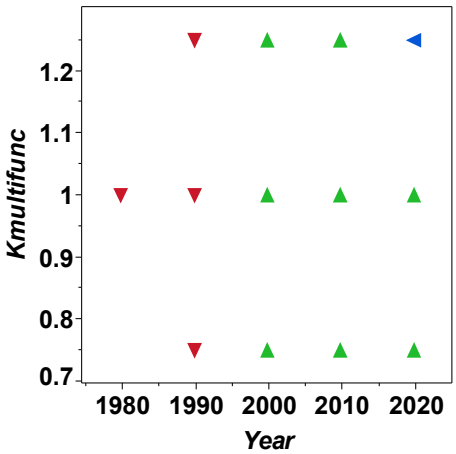


Figure 44. Sensitivity plot for $K_{multifunctional}$, $K_{motor} = K_{battery} = 1$

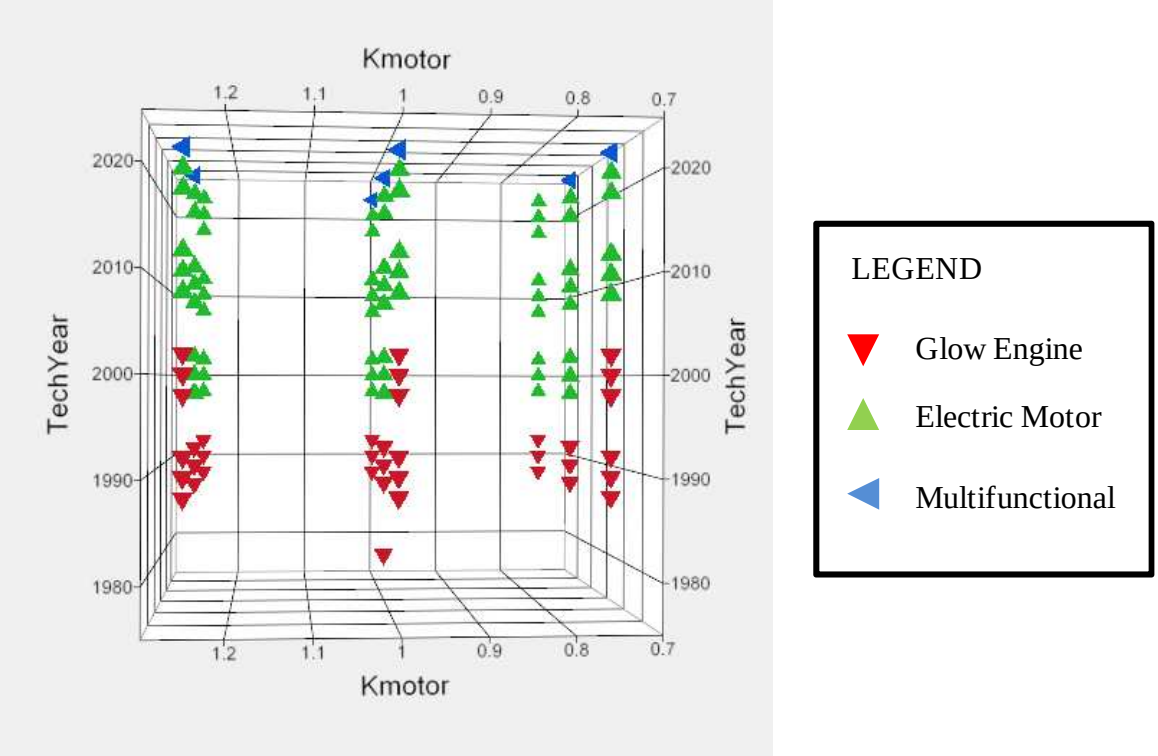


Figure 45. Architecture results for test case in quasi-four dimensions

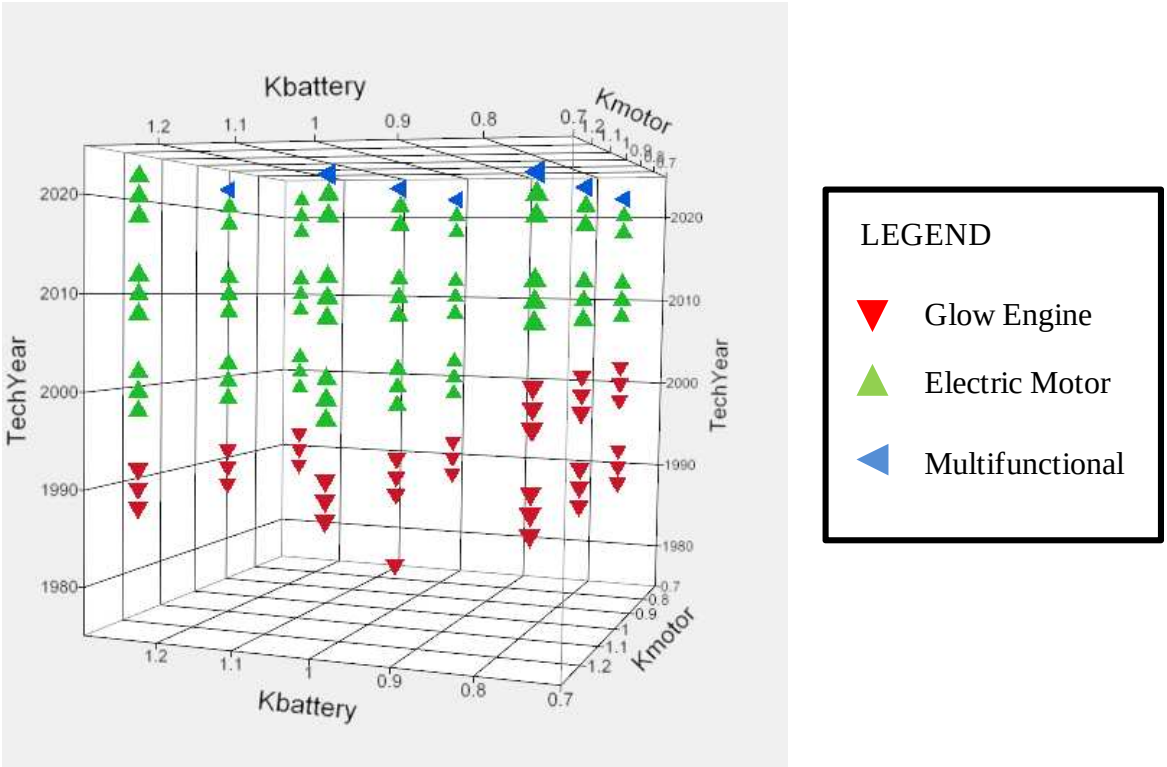


Figure 46. Architecture results for test case in quasi-four dimensions, rotated

As with any strategic tool, the results of this test case should be used not to conclude that all of the answers are now known, but to decide which questions are more important than others. An example of the result of this test case would be the development of a wait-and-see approach that identifies technical characteristics of electrical components to watch. In this case, it may also be desirable to study the fundamentals of an electrical architecture more thoroughly.

The above considerations make it clear that the type of information provided with this test case would be useful to a firm in considering its rivals.

The full factorial design had 108 cases, plus one case to give a baseline in 1980 for examination. The two images are of the same plot, but rotated in order to give the reader a better view and ability to perceive depth.

Review of Hypotheses

With the various tests and investigations completed, it is time to revisit the hypotheses that guided them. These hypotheses answered a series of research questions that were raised while considering the original problem of understanding the future of architectural change. While answering them does not answer the entire domain of change-related problems, they do serve to make an academic contribution to the current body of knowledge.

Hypothesis #1

A genetic algorithm with mutation and crossover operators formulated in graph theory can find an optimum in the search space of possible architectures by evaluating a small portion of them.

The support for this hypothesis can be found by considering the results of the combinatorial test space experiments. These experiments led to the conclusion that rectification and random completion operators should be included in the genetic

algorithm within Sindri. Further, it was decided that random completion should be applied to S'-Graphs at every generation of the genetic algorithm, not simply the beginning. Doing this led to the operators being applied in the following order: mutation, random completion, crossover, rectification, anti-rectification, and selection.

The first three design spaces should varying levels of performance. The best settings for the first design space led to an optimal result being found within a number of function calls that represented about 20% of the design space. The second design space appeared to be a rather easy one to traverse, and so required only about 3% of the total space to be examined for convergence. The final test space, however, did not reach an optimum even with enough function calls to perform an exhaustive search. This result was found before random completion was used in each generation. Once random completion is utilized at each generation, it appears that an optimum can be found in this space with about 30-40% of the total space.

Putting all of this together provides support for the first hypothesis, although it is important to consider what the word "small" is supposed to mean. In this case, "small" will be greatly relaxed to mean a minority of the combinatorial space is searched. However, the results above suggest that future work may be required in order to accelerate the algorithm. Suggestions for this are made two chapters from here.

Hypothesis #2

Using the selection operator to screen out nonsensical designs will be effective and cause minimal performance degradation, provided the program begins with well-connected parent solutions.

This hypothesis has been rejected. When the main effects data were reviewed for the test combinatorial spaces, it became apparent that the source-sink rectification operator made a major difference in the performance of the algorithm. Since the effects dealt with the means of given sets of response data, the means included both well-

connected and not well-connected starts. This means that well-connectedness had little effect on the increase in performance due to the inclusion of rectification.

The significance of the rectification operator to this hypothesis is that it serves to make sensible architectures out of poorly connected ones. Since changing poorly-connected architectures into well-connected ones markedly improves the genetic algorithm's performance, then the hypothesis must be false.

The rejection of this hypothesis also serves to guide the philosophical base of the genetic algorithm within Sindri. When it was first formulated, the philosophical stance was to minimize interference with the directed random search. It appears that this stance is too hands-off. This finding is important academically in future searches for optimizers and searches of the combinatorial spaces that represent architectural problems.

Hypothesis #3

If there is no initial architecture, the algorithm may start with a random selection of components. Further, if these components are assigned random (feasible) connections, it will greatly improve the performance of the algorithm.

This hypothesis has some support from the main effects analysis of the test combinatorial spaces. The two test spaces that deal with a single edge type show a major improvement in performance when the random connection flag is on. The final test space does not, but this may be due to the far larger number of functional calls that where performed, meaning that the initial performance boost is obscured by longer-term behavior of the optimization.

Philosophically speaking, the support of this hypothesis indicates that it is better to start with an initial baseline architecture (or even multiple baselines to be analyzed simultaneously or in series) than to leave the random process entirely in charge. This also makes some sense with the observed behavior of the genetic algorithm, namely that it finds convergence faster for solutions that are closer to its starting point.

Hypothesis #4 *Of the three methods surveyed here (hyperplanes, multidimensional S-curves, TFDEA), TFDEA will be seen to be the most accurate.*

This hypothesis has been refuted in the sense that there was no clear winner on the numerical comparison between TFDEA and multidimensional S-curves. In fact, multidimensional S-curves appeared to be more accurate over the test data that were applied to these methods, even though the difference was not statistically significant. More data and testing will be necessary to separate these two methods on a basis of accuracy if this is in fact possible.

The benefit of evidence applied to this hypothesis is to guide the choice of extrapolation model for future users of this method. It will also help to guide the efforts of those that wish to draw inspiration from this method and hybridize it with other approaches, or simply to utilize parts of it to solve sub-problems posed within the body of this dissertation.

After looking at the results of the hypotheses, it becomes apparent that the method of using a genetic algorithm as the combinatorial solver for Sindri provides some success in solving the problems with which it is presented. The hypotheses as written are the product of a stance that the random qualities of the genetic algorithm are best used without interference. This posture turned out to be rather incorrect when working with edges and connections between components. The results of this work strongly suggest a need for the use of more structure in order to improve performance and reduce the number of function calls wasted on nonsensical candidate architectures.

Investigating these hypotheses also provided useful information on the mutation operator and its best use. The random addition and subtraction of edges is somewhat superfluous when the use of the rectification operator generates many edges at a time and

the anti-rectification operator moves them around.

While this is not a direct outcome of investigating the hypotheses, another observation was made while working with the combinatorial test space and the helicopter test case. The behavior of the genetic algorithm paired with the random connection operator varies greatly with the generality or the specificity with which connection types are defined. If there are but one or two edge types, then there are many different arrangements for any given set of component vertices. Thus, a mutation operator tuned to swap connections around more than change the number of vertices being used may see good performance. On the other hand, if each component can only connect to one other due to a specific connection type, the problem is a simpler matter of chosen the appropriate vertex set to be connected. In this case, the algorithm would do better to highly favor vertex addition and removal, and allow the random connection operator to do the work of putting them together.

CHAPTER 6

CONCLUSIONS

It is time to revise the original problem that inspired this thesis. The problem was to get a better strategic understanding of systems development by gaining insights into the timing of potential shifts in dominant architectures in a given realm. These shifts are often enabled by a small number of technologies that either cause the architecture to deliver superior performance, or deliver adequate performance while satisfying other needs. In this case, the inspiration for the problem was the MAST research project, and so this led to the selection of a test case.

Multiple hypotheses on the expected behavior of a genetic algorithm approach to finding optimal architectures were formulated and tested. The results of these tests were to establish that a "pure" genetic algorithm (in the sense that only discrete edges and vertices are added or modified) is not sufficient for an efficient search of the combinatorial space presented by architectural graphs. The addition of random completion and rectification operators increased the efficiency of the base genetic algorithm. The behavior of the genetic algorithm also indicates that further structuring of the search algorithm may be worth pursuing.

A combined combinatorial and continuous optimizer approach was taken to a test case for small air vehicles. This approach was successful in finding sensible transition points between architectures and quantifying the evolution of a key parameter that these architectures have been attempting to satisfy. The visualization of the results also provided a strategic view by showing both the expected time of a new architecture's emergence, and the sensitivity of that time to changes of the rates of progress in multiple components simultaneously. The fact that these sensitivities were taken simultaneously helps to show were the relative strengths of competitors delay or accelerate given

outcomes.

The academic contribution of this research is to take a first step in a generalized, automated search algorithm between architectures that can operate without a strict functional assignment of components. This frees the designer to investigate the breakdown of functional boundaries as is required when considering trade-offs between modularity and synergy in design.

The search algorithm is the locus of the academic contribution. In addition to demonstrating its feasibility, the research serves to provide insight on appropriate parameters to apply to the search algorithm. In this case, there is an emphasis placed upon ensuring that every architecture that is generated meets some basic criteria for feasibility. This can be accomplished while maintaining an unbiased search by utilizing random finishing of the architecture.

In addition, the research presented in this dissertation suggests a series of new avenues to pursue in improving both the search algorithm and the graph representation of an architecture. Several ideas for furthering this work are presented in the next chapter.

CHAPTER 7

FUTURE WORK / SINDRI 2.0 DESCRIPTION

Improved Data Structure

In the current version of Sindri, the graph is encoded as a simple string of values, with vertices listed in no particular order other than that the sink is at the end of the list and the source is second to last. While developing the crossover operator, it became apparent that there was a preferred form for the chromosome. That form is the depth-first search tree, which arranges vertices into a tree based on how closely they are connected to the tree root. The permanent source vertex is a good choice for the root, since by definition edges can only leave it.

The depth-first search structure is a logical choice because of the features in a graph that it highlights. The DFS is the start point in multiple algorithms to find bridges, articulation points, and cycles. All of these elements are important in deciding how to create a more flexible crossover operator that can satisfy the assortment principle. Further, the ability to find and localize cycles is necessary to understand how to resolve the various transformations that the graph represents, in the same way that feedbacks within a Design Structure Matrix are necessary to understand in order to solve coupled analysis problems.

Adverse System Interactions

One of the original design ideas for Sindri that was inspired by TRIZ Substance-Field analysis was the ability to use the graph structure to quantify adverse system interactions. Examples of adverse interactions include electromagnetic fields from high power and electrical motors, vibration due to rocket engines, and excessive heat production from a wide variety of devices. The original design idea for Sindri was to

have the environment nodes, the source and sink, serve as the facilitators for accounting for this effect. However, there turned out to be no good way to separate the adverse effect accounting and the role of these vertices in modeling the resources available from the environment for the various architecture components to utilize.

The next iteration of design of the graph structure for the architecture would incorporate this feature by adding a new node. This node, the "near-field" node, would have a specialized logic associated with it in the same way that the source and sink vertices do. This logic would require it to accept any physical quantity (magnetism, heat, etc.) that is sent to it, and distribute it to all vertices that were susceptible to it. There may even need to be a new class of input/output port to separate this from the "designer intended" connections between vertices.

Once this vertex type and its logic were implemented, the genetic algorithm within Sindri would be given an incentive (through a better final objective value) to either avoid components that produce these adverse effects or to somehow isolate them. The use of compensators, of course, may also impose other penalties upon the system. This would properly emulate the situation of physical systems, where new architectures must often compensate for some internal problem. A major surge in improvement can result once this hindrance is eliminated in a more elegant fashion than simple compensation.

Improved Crossover

One method to improve the improve the performance of the algorithm would be to make the crossover operator more general. The current operator will only sever the graph at bridges. However, it is possible to sever the graph at many other locations, including complimentary ends of cycles by splitting a ring in half in both parent graphs, and then reconnecting the two halves. It may also be possible to devise a way to create "interior" and "exterior" splits by finding compatible pairs of edges (those with the same type) in two graphs and then using them to reconnect paired fragments of graphs.

In any case, having a more generalized crossover operator would allow for more general "subsystems" of a given architecture to be swapped between parents and far rapid traversal of the design space.

Better Classification of Vertex Classes

The experience gained from working with the case study indicates a need for multiple new ways of looking at vertices. At the start of this work, there were simply two types of vertices: the component vertices and vertices to represent the environment. There appears to be a need for many more, and to move them into specific roles. A new classification of vertices to implement should include the following:

- "Far" environment that clean resources can be obtained from and wastes can be expelled to without interfering with the system
- "Near" environment, which accounts for outputs of components that cannot be easily pushed away from the system and may interfere with other components
- Transformation components, which take in a physical quantity and either increase (decrease) it or change it into another physical quantity
- Tank components, which store either materials to be utilized (e.g., fuel) or products that cannot be vented from the system (e.g., used reagents), and are often the limiting factor in endurance and mission performance analyses
- Connector components, for structures or power distribution, which serve to link together a common load (e.g., forces, electrical current) from a large number of other components in the system

Structure for Constraint and Mission Analysis

The graph structure of Sindri provides for an easy pathway for developing Design Structure Matrices for any physical architecture. The inputs and outputs from the components that are combined into the architecture are also available for analysis for the

best way to evaluate the system. Thus, there are a variety of ways to handle the continuous variables available in the graph in addition to using an optimizer directly.

For a more thorough evaluation of a given architecture, various conditions should be applied to perform sizing by both constraint and mission analysis. The easiest to do of these is constraint analysis, and should be the first to be computed. At first blush, it is a simple matter of applying multiple conditions to the architecture and making sure the constraints are satisfied by the design. However, there is a coupling between sizing due to constraints and sizing due to the mission.

In the case of flying vehicles, the weight is often coupled between constraints (e.g., minimal accelerations, or the need to generate enough lift to fly) and the mission (i.e., sufficient fuel). In conceptual design, the constraint analysis is often handled by dealing with the constraints in normalized quantities such as thrust to weight. This is a worthwhile approach, but it needs to be generalized if it is to used in an automated program. This would be a worthwhile avenue of research to pursue in the context of evaluating architectures that have been developed automatically by a master program.

Summary

The current version of Sindri and the research presented in this dissertation show the way to multiple new directions. Like any good investigation, just as many new questions are raised as answered. There are areas that have been targeted for planned improvements to enhance the stability and effectiveness of the code. Beyond an update to the data structures and organization of components, there are multiple areas where greater study would be fruitful. The main thrusts for new research are to enhance the methods for solving the continuous problem and to deal more thoroughly with system-wide effects.

APPENDIX A: STATE-OF-THE-ART CURVE COMPARISON DATA

This appendix contains a series of data sets that were used to evaluate the accuracy of multiple models for technology forecasting. Each of these datasets were taken from the existing literature. The data sets varied in the number of characteristics to be fit to the model and the relative rates of change. The originals of these data sets can be found by following the references identified in the captions for the tables.

Table 10. Jet engines historical dataset

Engine Name	Year Introduced	SFC (lb_f/lb_m*hr)	Thrust (lb_f)
JT9D-7J	1970	0.665	50000
CF6-6D	1970	0.646	40000
JT9D-7	1971	0.665	46300
CF6-50A	1971	0.656	49000
RB211-22B	1972	0.628	42000
JT9D-7A	1972	0.663	46950
CF6-50C	1973	0.657	51000
CF6-50E	1973	0.657	52500
JT9D-59A	1974	0.646	53000
JT9D-70A	1974	0.646	53000
JT9D-7F	1974	0.665	48000
RB211-524B	1975	0.643	50000
CF6-50C1	1976	0.657	52500
CF6-50C2	1976	0.63	52500
CF6-50E2	1976	0.63	52500
CF6-45A2	1978	0.63	46500
CF6-50C2-B	1978	0.63	54000
RB211-524C2	1979	0.642	51500
JT9D-7Q	1979	0.643	53000
JT9D-7R4D	1980	0.596	48000
RB211-524D4	1981	0.617	53000
RB211-535C	1981	0.646	37400
CF6-80A	1981	0.623	48000
CF6-80A2	1981	0.623	50000
JT9D-7R4E	1982	0.596	50000
JT9D-7R4G2	1982	0.599	54750
RB211-535E4	1983	0.598	40100
PW2000-037	1983	0.582	38400
CFM56-3B1	1984	0.661	20000
CFM56-2B	1984	0.657	22000

CF6-80C2A2	1985	0.578	53500
CF6-80C2B1	1985	0.576	57900
CFM56-2A	1986	0.66	24000
CF6-80C2B1F	1986	0.564	57900
CF6-80C2B2	1986	0.576	52500
PW2000-040	1987	0.582	41700
CF6-80C2B4	1987	0.576	57900
CF6-80C2B6	1987	0.58	60800
RB211-524G	1988	0.57	58000
CFM56-5A3	1988	0.596	26500
RB211-524H	1989	0.57	60600
CFM56-3C	1990	0.651	23500
CF6-80C2B6F	1992	0.58	60800
CFM56-5C4	1993	0.567	34000

Table 11. Microchip historical dataset

Developer	Year	SPEC95 Int	SPEC95 FP	Feature Size	Power	Package Size	Trans. Count
PA-RISC	1992	3.2	4	0.8	23	196	0.8
Alpha	1992	3	3.6	0.75	30	299	9.3
SPARC	1992	1.5	1.7	0.6	14.2	315	5.2
Alpha	1994	8.5	12.7	0.5	50	209	9.3
PA-RISC	1994	5.2	4.6	0.8	30	259	9
PA-RISC	1994	4.6	4.7	0.8	10	345	3.9
Alpha	1994	5.2	6.3	0.5	33	138	3.5
Intel	1994	3.3	2.8	0.6	5	148	3.1
Intel	1994	2.9	2.5	0.6	6.5	150	3.1
Intel	1995	8.2	6.8	0.35	35	195	5.5
Power PC	1995	6	6	0.5	30	67	6.35
Intel	1995	6.1	5.4	0.6	29.2	195	5.5
Intel	1995	3.8	3	0.35	10	90	3.1
SPARC	1995	7.8	11.4	0.42	30	156	5.4
SPARC	1995	6.6	9.4	0.47	30	149	5.4
Power PC	1995	4.7	3.8	0.5	5.6	311	7
SPARC	1995	3.5	3.5	0.6	15	265	5.2
SPARC	1995	1.6	2	0.4	9	299	3.1
Intel	1996	7.3	6.2	0.35	29.4	131	7.5
SPARC	1996	10.4	15	0.35	25	132	3
Power PC	1996	6.3	4.6	0.35	5.5	197	3.6
Intel	1996	4.8	3.3	0.35	10	90	3.1
MIPS	1996	5.5	5.5	0.35	10	596	3.43
SPARC	1996	3.5	3	0.35	9	162	8.8
MIPS	1996	11.4	19.1	0.35	30	196	0.85
Intel	1996	4.3	3.3	0.35	10	90	3.1
Alpha	1996	12.3	17.2	0.35	30.5	350	10
Power PC	1997	12	10	0.5	85	83	6.5

Alpha	1997	17.3	19.9	0.35	40.5	302	15.2
MIPS	1997	13	15	0.25	13	298	6.8
AMD	1997	6.2	3.5	0.35	20	184	22
Intel	1997	6.4	4.7	0.35	15.7	141	4.5
Intel	1997	7.1	5.2	0.35	17	141	4.5
AMD	1997	6.8	3.8	0.35	28.3	184	22
Cyrix	1997	5.1	2.2	0.35	10	162	8.8
Intel	1997	5.6	4.3	0.35	7.8	141	4.5
Intel	1997	11.9	8.6	0.35	43	131	7.5
Power PC	1997	12.9	8.5	0.25	12	150	2.7
Alpha	1998	44	66	0.35	72	78	1.4
SPARC	1998	14.2	16.9	0.35	30	197	6
Power PC	1998	12.2	7.1	0.25	4.2	234	1.68
Intel	1998	11.9	8.6	0.25	17.5	131	9.5
Intel	1998	11.9	8.6	0.25	26	131	7.5
Intel	1998	16.5	13.7	0.25	23.3	131	7.5
Intel	1998	18.9	14.7	0.25	23.3	131	7.5
AMD	1999	23.6	20.6	0.25	46	164	1.2
AMD	1999	31.7	24	0.25	50	198	2.4
Intel	1999	35.6	30.4	0.18	34	106	28
Intel	1999	15.1	11.8	0.25	23.4	225	19
Intel	1999	24	15.9	0.25	34	106	9.5
Intel	1999	17.9	12.9	0.25	23.4	225	19
Power PC	1999	21.4	20.4	0.15	8	164	2.8

Table 12. Jet fighter historical dataset

Aircraft	First Flight	Max Mach	MFTF	Payload	BVR range
F80	1944	0.65	1	500	0
F84	1946	0.8	1	366	0
F86	1947	0.9	1	480	0
F89	1949	0.75	0.5	500	5
F94	1950	0.7	0.5	500	0
F100	1953	1.4	0.5	500	0
F101	1954	1.6	0.5	1000	6
F102	1953	1.2	0.5	1000	6
F104	1954	2	0.5	1000	6
F106	1955	1.8	0.7	1000	0
F8	1956	2	0.6	1000	6
F5A	1959	1.4	2.5	706	0
F4E	1967	2.2	1.1	1000	10
F14	1971	2	0.7	2000	30
F5E	1971.5	1.6	3	5000	0
Fl5	1972	2.4	2	2000	20
F16	1974	1.8	2.4	3000	0
F18	1978	1.8	3.2	3000	20
F20	1982	2	4	3000	20

APPENDIX B: COMBINATORIAL SPACE GENERATOR CODE LISTING

The following Visual Basic for Applications (VBA) computer program may be useful in developing test functions for combinatorial optimizers / search algorithms such as the one used in this dissertation. This program will generate the combinatorial space of acyclic directed graphs with labeled inputs and outputs, functional maps between those inputs and outputs, and limit the graphs that are well-connected. It is a recursive program, and so builds up the graph vertex by vertex, searching the combinatorial space of possible connections for each combination of vertices that are generated.

```
Sub Populate_SetA()

    ' Generate all of the possible combinations of nodes
    Dim functions(100) As String
    Dim chain(990000, 50) As String
    Dim currentIndex As Integer
    Dim currentCount As Long
    Dim stopIndex As Integer
    Dim functionsCount As Integer
    Application.Calculation = xlCalculationManual
    maxNumberHits = 2
    done = False
    counter = 2
    functionsCount = 0

    Do While done = False
        If Cells(counter, 4).value = "A" Then
            functions(counter - 2) = Cells(counter, 1).value
            functionsCount = functionsCount + 1
        End If
        counter = counter + 1
        If Cells(counter, 1).value = "" Or Cells(counter, 1).value = Empty Then
            done = True
        End If
    Loop
    currentCount = 0
    counter = 0
```

```
   currentIndex = 0
   For j = 1 To functionsCount * maxNumberHits  'number of modules to add in
      stopIndex = j
      stepOut = recursiveBuild(chain, currentIndex, stopIndex, currentCount,
functionsCount, functions)
   Next
      ' once the set is built, populate the proper page
   Sheets("SetA").Select
   For i = 0 To currentCount - 1
      Cells(i + 4, 1).Formula = "=$B$1"
      sideCount = 0
      weit = ""
      Do While chain(i, sideCount) <> ""
         If sideCount > 0 Then
            weit = weit & " + "
         End If
         insertNo = Trim(Str(findFunctions(functions, chain(i, sideCount))))
         Cells(i + 4, sideCount + 2).Formula = "=" & chain(i, sideCount) & _
            "(" & Cells(i + 4, sideCount + 1).Address & ", FunctionalTypeCatalogue!F" &
            insertNo & ", FunctionalTypeCatalogue!G" & insertNo & ",
FunctionalTypeCatalogue!H" & _
            insertNo & ")"
         Cells(i + 4, sideCount + 42).value = chain(i, sideCount)

         sideCount = sideCount + 1
         weit = weit & "FunctionalTypeCatalogue!E" & insertNo
      Loop
      weightString = "=" & Cells(i + 4, sideCount + 1).Address & " / (" & _
         weit & ")"
      Cells(i + 4, 40).Formula = "=" & Cells(i + 4, sideCount + 1).Address
      Cells(i + 4, 41).value = weightString
   Next
   Application.Calculation = xlCalculationAutomatic
End Sub

Function recursiveBuild(chain() As String, currentIndex As Integer, stopIndex As
Integer, ByRef currentCount As Long, _
   functionCount As Integer, functions() As String)
   maxNumberHits = 2
   If currentIndex = stopIndex Then 'reached the maximum depth of recursion
      currentCount = currentCount + 1
      If currentCount = 274 Then
         teststop = 1
      End If
      recursiveBuild = stopIndex - 1
      Exit Function
```

```
    End If
    stepOutIndex = 0
    For i = 1 To functionCount
        overload = False
        If i > 1 And chain(currentCount, 0) = "" Then
            For m = 1 To currentIndex
                chain(currentCount, m - 1) = chain(currentCount - 1, m - 1)
            Next
        End If
        If currentIndex < stepOutIndex Then 'clean up for new entries
            For m = currentIndex + 1 To stopIndex
                chain(currentCount, m - 1) = ""
            Next
        End If
        hitsCount = 0
        For j = 0 To 49
            If chain(currentCount, j) = functions(i - 1) Then
                hitsCount = hitsCount + 1
            End If
            If hitsCount >= maxNumberHits Then
                overload = True
                Exit For
            End If
        Next
        If overload = True Then
            'do nothing
        Else
            chain(currentCount, currentIndex) = functions(i - 1)
            stepOutIndex = recursiveBuild(chain, currentIndex + 1, stopIndex, currentCount,
functionCount, functions)
        End If
    Next
    recursiveBuild = currentIndex
End Function
Function findFunctions(functions, functionName)
    findFunctions = -1
    For i = 1 To 100
        If functions(i - 1) = functionName Then
            findFunctions = i + 1
            Exit For
        End If
    Next
End Function

Sub Populate_SetB()
' Generate all of the possible combinations of nodes
```

```
    Dim functions(100) As String
    Dim vars(100, 2) As Integer
    Dim chain(999900, 50) As String
    Dim links(999900, 50) As String
    Dim forwardArray(20) As Integer
    Dim oldChain(50) As String
    Dim currentIndex As Integer
    Dim currentCount As Long
    Dim stopIndex As Integer
    Dim functionsCount As Integer
    Application.Calculation = xlCalculationManual
    maxNumberHits = 1

    done = False
    counter = 2
    functionsCount = 0
    Do While done = False
        If Cells(counter, 4).value = "A" Or Cells(counter, 4).value = "B" Then
            functions(counter - 2) = Cells(counter, 1).value
            functionsCount = functionsCount + 1
            vars(counter - 2, 0) = Cells(counter, 27).value
            vars(counter - 2, 1) = Cells(counter, 28).value
        End If
        counter = counter + 1
        If Cells(counter, 1).value = "" Or Cells(counter, 1).value = Empty Then
            done = True
        End If
    Loop
    currentCount = 0
    counter = 0
    currentIndex = 0
    For j = 1 To 8 'functionsCount * maxNumberHits  'number of modules to add in
        stopIndex = j
        stepOut = recursiveBuildWLinker(chain, currentIndex, stopIndex, currentCount,
functionsCount, functions, links, vars)
    Next
    ' once the set is built, populate the proper page
    'attempt to import text file info
    startCell = 4
    Sheets("SetB").Select
    For i = 0 To currentCount - 1
        ' use the links to get the appropriate formulas written
        For m = 0 To 20
            forwardArray(m) = m
        Next
        Cells(i + startCell, 1).Formula = "=$B$1"
```

```
        sideCount = 0
        weit = ""
        'can assume that flavors progress from 0 to 2 for each start node
        Do While chain(i, sideCount) <> ""
            flavorNo = 0
            If sideCount > 0 Then
                weit = weit & " + "
            End If
            insertNo = Trim(Str(findFunctions(functions, chain(i, sideCount))))
            If vars(insertNo - 2, 1) = 2 And InStr(chain(i, sideCount), "_") = 0 Then
                'compensate for previous expansions
                compensator = 0
                For sc2 = 0 To sideCount
                    If sc2 > 0 Then
                        If InStr(chain(i, sc2), "_") > 0 And InStr(chain(i, sc2 - 1), "_") > 0 Then
                            compensator = compensator + 1
                        End If
                    End If
                Next
                'expand the multiple outputs
                popUp1 = chain(i, sideCount) & "_2"
                chain(i, sideCount) = chain(i, sideCount) & "_1"
                counter = sideCount
                Do While popUp1 <> ""
                    forwardArray(counter + 1 - compensator) = forwardArray(counter + 1 -
compensator) + 1
                    popUp2 = chain(i, counter + 1)
                    chain(i, counter + 1) = popUp1
                    popUp1 = popUp2
                    counter = counter + 1
                Loop
            End If
            If Sheets("FunctionalTypeCatalogue").Cells(insertNo, 4) = "A" Then
                downscanner = 0
                    Do While links(i, downscanner) <> ""
                        strip = Split(links(i, downscanner), ":")
                        numbers = Split(links(i, downscanner), ".")
                        If InStr(strip(0), functions(insertNo - 2)) > 0 And
forwardArray(Int(Trim(numbers(0)))) = sideCount Then
                            input1string = strip(1)
                            scanner2 = 0
                            If strip(1) = "Source" Then
                                input1set = Cells(i + 4, 1).Address
                            Else
                                Do While chain(i, scanner2) <> ""
                                    tempA = Split(chain(i, scanner2), "_")
```

```
                        chainString = tempA(0)
                        If InStr(input1string, chainString) > 0 Then
                            localeA = Split(input1string, ".")
                            locale = localeA(0)
                            locale = forwardArray(locale)
                            input1set = Cells(i + startCell, 2 + locale).Address
                            scanner2 = scanner2 + 1
                            Exit Do
                        End If
                        scanner2 = scanner2 + 1
                    Loop
                    Exit Do
                End If
            End If
            downscanner = downscanner + 1
        Loop
      topString = "=" & chain(i, sideCount) & _
        "(" & input1set & _
        ", FunctionalTypeCatalogue!F" & _
        insertNo & ", FunctionalTypeCatalogue!G" & insertNo & ", FunctionalTypeCatalogue!H" & _
        insertNo & ")"
      Cells(i + startCell, sideCount + 2).Formula = topString
      Cells(i + startCell, sideCount + 42).value = chain(i, sideCount)
      sideCount = sideCount + 1
    ElseIf Sheets("FunctionalTypeCatalogue").Cells(insertNo, 4) = "B" Then
      If vars(insertNo - 2, 0) = 1 Then
        downscanner = 0
        Do While links(i, downscanner) <> ""
          strip = Split(links(i, downscanner), ":")
          numbers = Split(links(i, downscanner), ".")
          If InStr(strip(0), functions(insertNo - 2)) > 0 And forwardArray(Int(Trim(numbers(0)))) = sideCount Then
            input1string = strip(1)
            scanner2 = 0
            If strip(1) = "Source" Then
              input1set = Cells(i + startCell, 1).Address
              downscanner = downscanner + 1
              Exit Do
            Else
              Do While chain(i, scanner2) <> ""
                tempA = Split(chain(i, scanner2), "_")
                chainString = tempA(0)
                If InStr(input1string, chainString) > 0 Then
                  localeA = Split(input1string, ".")
                  locale = localeA(0)
```

```
                        oldLocale = locale
                        locale = forwardArray(locale)
                        booster = 0
                        If forwardArray(oldLocale + 1) - forwardArray(oldLocale) > 1
And strip(2) = "vanilla" Then
                            booster = 1
                        End If
                        input1set = Cells(i + startCell, 2 + locale + booster).Address
                        scanner2 = scanner2 + 1
                        downscanner = downscanner + 1
                        Exit Do
                    End If
                    scanner2 = scanner2 + 1
                Loop
                Exit Do
            End If
        End If
        downscanner = downscanner + 1
    Loop
    input2set = ""
ElseIf vars(insertNo - 2, 0) = 2 Then
    downscanner = 0
    Do While links(i, downscanner) <> ""
        strip = Split(links(i, downscanner), ":")
        numbers = Split(links(i, downscanner), ".")
        If InStr(strip(0), functions(insertNo - 2)) > 0 And
forwardArray(Int(Trim(numbers(0)))) = sideCount Then
            inputAstring = strip(1)
            scanner2 = 0
            If strip(1) = "Source" Then
                inputAset = Cells(i + startCell, 1).Address
                downscanner = downscanner + 1
                Exit Do
            Else
                Do While chain(i, scanner2) <> ""
                    tempA = Split(chain(i, scanner2), "_")
                    chainString = tempA(0)
                    If InStr(inputAstring, chainString) > 0 Then
                        localeA = Split(inputAstring, ".")
                        locale = localeA(0)
                        oldLocale = locale
                        locale = forwardArray(locale)
                        booster = 0
                        If forwardArray(oldLocale + 1) - forwardArray(oldLocale) > 1
And strip(2) = "vanilla" Then
                            booster = 1
```

```
                        End If
                        inputAset = Cells(i + startCell, 2 + locale + booster).Address
                        scanner2 = scanner2 + 1
                        Exit Do
                    End If
                    scanner2 = scanner2 + 1
                Loop
                Exit Do
            End If
        End If
        downscanner = downscanner + 1
    Loop
    If strip(2) = "chocolate" Then
        input1set = inputAset
        flavorNo = flavorNo + 1
    Else
        input2set = inputAset
    End If
    If flavorNo = 1 Then
        checkFlavor = "vanilla"
    Else
        checkFlavor = "chocolate"
    End If
    Do While links(i, downscanner) <> ""
        strip = Split(links(i, downscanner), ":")
        numbers = Split(links(i, downscanner), ".")
        If InStr(strip(0), functions(insertNo - 2)) > 0 And
forwardArray(Int(Trim(numbers(0)))) = sideCount And strip(2) = checkFlavor Then
            inputBstring = strip(1)
            scanner2 = 0
            If strip(1) = "Source" Then
                inputBset = Cells(i + startCell, 1).Address
                downscanner = downscanner + 1
                Exit Do
            Else
                Do While chain(i, scanner2) <> ""
                    tempA = Split(chain(i, scanner2), "_")
                    chainString = tempA(0)
                    If InStr(inputBstring, chainString) > 0 Then
                        localeA = Split(inputBstring, ".")
                        locale = localeA(0)
                        oldLocale = locale
                        locale = forwardArray(locale)
                        booster = 0
                        If forwardArray(oldLocale + 1) - forwardArray(oldLocale) > 1
And strip(2) = "vanilla" Then
```

```
                booster = 1
            End If
            inputBset = Cells(i + startCell, 2 + locale + booster).Address
            scanner2 = scanner2 + 1
            Exit Do
          End If
          scanner2 = scanner2 + 1
        Loop
        Exit Do
      End If
    End If
    downscanner = downscanner + 1
  Loop
  If strip(2) = "chocolate" Then
    input1set = inputBset
  Else
    input2set = inputBset
  End If
End If
If input2set <> "" Then
functionString = chain(i, sideCount)
topString = "=" & functionString & _
  "(" & input1set & ", " & input2set & _
  ", FunctionalTypeCatalogue!F" & _
  insertNo & ", FunctionalTypeCatalogue!G" & insertNo & ",
FunctionalTypeCatalogue!H" & _
  insertNo & ", FunctionalTypeCatalogue!I" & insertNo & ",
FunctionalTypeCatalogue!J" & _
  insertNo & ", FunctionalTypeCatalogue!K" & insertNo & ")"
Cells(i + startCell, sideCount + 2).Formula = topString
Cells(i + startCell, sideCount + 42).value = chain(i, sideCount)
sideCount = sideCount + 1
End If
If vars(insertNo - 2, 0) = 1 And vars(insertNo - 2, 1) = 2 Then
  functionString = chain(i, sideCount)
  topString = "=" & functionString & _
    "(" & input1set & _
    ", FunctionalTypeCatalogue!F" & _
    insertNo & ", FunctionalTypeCatalogue!G" & insertNo & ",
FunctionalTypeCatalogue!H" & _
    insertNo & ", FunctionalTypeCatalogue!I" & insertNo & ",
FunctionalTypeCatalogue!J" & _
    insertNo & ", FunctionalTypeCatalogue!K" & insertNo & ")"
  Cells(i + startCell, sideCount + 2).Formula = topString
  Cells(i + startCell, sideCount + 42).value = functionString
  sideCount = sideCount + 1
```

```
            functionString = chain(i, sideCount)
            topString = "=" & functionString & _
               "(" & input1set & _
               ", FunctionalTypeCatalogue!F" & _
               insertNo & ", FunctionalTypeCatalogue!G" & insertNo & ",
FunctionalTypeCatalogue!H" & _
               insertNo & ", FunctionalTypeCatalogue!I" & insertNo & ",
FunctionalTypeCatalogue!J" & _
               insertNo & ", FunctionalTypeCatalogue!K" & insertNo & ")"
            Cells(i + startCell, sideCount + 2).Formula = topString
            Cells(i + startCell, sideCount + 42).value = functionString
            sideCount = sideCount + 1
         End If
      End If
      weit = weit & "FunctionalTypeCatalogue!E" & insertNo
   Loop
   weightString = "=" & Cells(i + 4, sideCount + 1).Address & " / (" & _
      weit & ")"
   Cells(i + startCell, 40).Formula = "=" & Cells(i + 4, sideCount + 1).Address
   Cells(i + startCell, 41).value = weightString
 Next
 Application.Calculation = xlCalculationAutomatic
End Sub
Function recursiveBuildWLinker(chain() As String, currentIndex As Integer, stopIndex
As Integer, ByRef currentCount As Long, _
 functionCount As Integer, functions() As String, links() As String, vars() As Integer)
 maxNumberHits = 1
 If currentIndex = stopIndex Then 'reached the maximum depth of recursion
   ' do link analysis here to make connections
   Call linker(chain, stopIndex, links, currentCount, vars, functionCount, functions)

   currentCount = currentCount + 1
   recursiveBuildWLinker = stopIndex - 1
   Exit Function
 End If
 stepOutIndex = 0
 For i = 1 To functionCount
   overload = False
   If i > 1 And chain(currentCount, 0) = "" Then
      For m = 1 To currentIndex
         chain(currentCount, m - 1) = chain(currentCount - 1, m - 1)
      Next
   End If
   If currentIndex < stepOutIndex Then 'clean up for new entries
      For m = currentIndex + 1 To stopIndex
         chain(currentCount, m - 1) = ""
```

```
      Next
    End If
    hitsCount = 0
    For j = 0 To 49
      If chain(currentCount, j) = functions(i - 1) Then
        hitsCount = hitsCount + 1
      End If
      If hitsCount >= maxNumberHits Then
        overload = True
        Exit For
      End If
    Next
    If overload = True Then
      'do nothing
    Else
      chain(currentCount, currentIndex) = functions(i - 1)
      stepOutIndex = recursiveBuildWLinker(chain, currentIndex + 1, stopIndex,
currentCount, functionCount, functions, links, vars)
    End If
  Next
  recursiveBuildWLinker = currentIndex
End Function
Sub linker(chainSet() As String, length As Integer, linkSet() As String, ByRef
currentCount As Long, vars() As Integer, _
      functionCount As Integer, functions() As String)
  Dim functionVector(50)
  Dim inPortsOpen(50, 2)
  Dim outPortsOpen(50, 2)
  Dim linkPairs(500, 2) As Integer
  Dim lastTap(50) As Integer
  ' MAJOR ASSUMPTION - each input and output are of different types:
  ' Port 1 - chocolate   - Block A
  ' Port 2 - vanilla     - Block B
  ' Port 3 - strawberry  - Block C
  ' UNLESS - only one open or closed port, then flavor matches that of block above
  ' look at the current chainSet to develop the first cut of links
  counter = 0
  cakeLayers = 2
  For i = 0 To length - 1
    For j = 0 To functionCount - 1
      If chainSet(currentCount, i) = functions(j) Then
        functionVector(i) = j
        Exit For
      End If
    Next
    ' set up input ports
```

```
    If vars(functionVector(i), 0) = 1 Then
        If Sheets("FunctionalTypeCatalogue").Cells(functionVector(i) + 2, 4).value = "A"
Then
            inPortsOpen(i, 0) = True
            inPortsOpen(i, 1) = False
            inPortsOpen(i, 2) = False
        ElseIf Sheets("FunctionalTypeCatalogue").Cells(functionVector(i) + 2, 4).value =
"B" Then
            inPortsOpen(i, 0) = False
            inPortsOpen(i, 1) = True
            inPortsOpen(i, 2) = False
        ElseIf Sheets("FunctionalTypeCatalogue").Cells(functionVector(i) + 2, 4).value =
"C" Then
            inPortsOpen(i, 0) = False
            inPortsOpen(i, 1) = False
            inPortsOpen(i, 2) = True
        End If
    ElseIf vars(functionVector(i), 0) = 2 Then
        inPortsOpen(i, 0) = True
        inPortsOpen(i, 1) = True
        inPortsOpen(i, 2) = False
    ElseIf vars(functionVector(i), 0) = 3 Then
        inPortsOpen(i, 0) = True
        inPortsOpen(i, 1) = True
        inPortsOpen(i, 2) = True
    End If
    ' set up output ports
    If vars(functionVector(i), 1) = 1 Then
        If Sheets("FunctionalTypeCatalogue").Cells(functionVector(i) + 2, 4).value = "A"
Then
            outPortsOpen(i, 0) = True
            outPortsOpen(i, 1) = False
            outPortsOpen(i, 2) = False
        ElseIf Sheets("FunctionalTypeCatalogue").Cells(functionVector(i) + 2, 4).value =
"B" Then
            outPortsOpen(i, 0) = False
            outPortsOpen(i, 1) = True
            outPortsOpen(i, 2) = False
        ElseIf Sheets("FunctionalTypeCatalogue").Cells(functionVector(i) + 2, 4).value =
"C" Then
            outPortsOpen(i, 0) = False
            outPortsOpen(i, 1) = False
            outPortsOpen(i, 2) = True
        End If
    ElseIf vars(functionVector(i), 1) = 2 Then
        outPortsOpen(i, 0) = True
```

```
            outPortsOpen(i, 1) = True
            outPortsOpen(i, 2) = False
        ElseIf vars(functionVector(i), 1) = 3 Then
            outPortsOpen(i, 0) = True
            outPortsOpen(i, 1) = True
            outPortsOpen(i, 2) = True
        End If
    Next
    startCount = currentCount
    ' an attempt at a better solution
    linkCount = 0
    ' enumerate all potential links
    edgeCounter = 0
    For m = 0 To 2
        For i = length - 1 To 0 Step -1
            For j = i - 1 To -1 Step -1
                If j = -1 Then
                    If inPortsOpen(i, m) = True Then
                    linkPairs(edgeCounter, 0) = i
                    linkPairs(edgeCounter, 1) = j
                    linkPairs(edgeCounter, 2) = m
                    edgeCounter = edgeCounter + 1
                    End If
                ElseIf inPortsOpen(i, m) = True And outPortsOpen(j, m) = True Then
                    linkPairs(edgeCounter, 0) = i
                    linkPairs(edgeCounter, 1) = j
                    linkPairs(edgeCounter, 2) = m
                    edgeCounter = edgeCounter + 1
                End If
            Next
        Next
    Next
    linkPairs(edgeCounter, 0) = -2 'give the endpoint
    linkPairs(edgeCounter, 1) = -2
    linkPairs(edgeCounter, 2) = -2
    If (currentCount = 7) Then
        teststop = 1
    End If
    For j = 0 To 50
        lastTap(j) = -2
    Next
    Call linkerRecursion(functionVector, linkSet, inPortsOpen, outPortsOpen, length,
functions, currentCount, linkCount, linkPairs, 0, 0, lastTap)
    endCount = currentCount 'for checking on results
    If endCount - startCount > 4 Then
        teststop = 1
```

```
End If
If endCount - startCount > 0 Then
   If endCount - startCount > 1 Then
      For y = startCount To endCount
         cleanOut = True
         For m = y + 1 To endCount
            n = 0
            Do While Not (linkSet(y, n) = "" And linkSet(m, n) = "")
               If linkSet(y, n) <> linkSet(m, n) Then
                  cleanOut = False
                  Exit Do
               End If
               n = n + 1
            Loop
            If cleanOut = True Then
               n = 0
               Do While linkSet(m, n) <> ""
                  linkSet(m, n) = ""
                  n = n + 1
               Loop
            End If
         Next
         If linkSet(y, 0) <> "" Then
            n = 0
            Do While chainSet(startCount, n) <> ""
               chainSet(y, n) = chainSet(startCount, n)
               n = n + 1
            Loop
         End If
      Next
      For y = startCount To endCount
         If linkSet(y, 0) = "" Then
            currentCount = currentCount - 1
            stepper = 0
            Do While linkSet(y + 1, stepper) <> ""
               linkSet(y, stepper) = linkSet(y + 1, stepper)
               stepper = stepper + 1
            Loop
         End If
      Next
   ElseIf linkSet(currentCount, 0) = "" Then
      currentCount = currentCount - 1
   End If
End If
If linkSet(currentCount, 0) = "" Then
   currentCount = currentCount - 1
```

```
   End If
   For y = startCount To currentCount
      If chainSet(y, 0) = "" And linkSet(y, 0) <> "" Then
         teststop = 1
      End If
   Next
End Sub

Function linkerRecursion(functionVector, linkSet, ByRef inPortsOpen, ByRef
outPortsOpen, length, functions, ByRef currentCount, ByRef linkCount, linkPairs,
startPoint, recurseLevel, lastTap)
   Dim passIn(50, 2)
   Dim passOut(50, 2)
   'return value of function is a flag showing direction of movement (return or advance)
   stepDirection = "advance"
   counter = startPoint

   ' we know that there is a walk from right to left in terms of available links
   Do While linkPairs(counter, 0) > -2
      If linkPairs(counter, 1) = -1 And inPortsOpen(linkPairs(counter, 0),
linkPairs(counter, 2)) = True Then
         ' if adding the source makes an identical to the previous, don't do it
         If stepDirection = "advance" Then 'only allow source connections on the
descending side
            If linkPairs(counter, 2) = 0 Then
               flavor = "chocolate"
            ElseIf linkPairs(counter, 2) = 1 Then
               flavor = "vanilla"
            ElseIf linkPairs(counter, 2) = 2 Then
               flavor = "strawberry"
            End If
            inPortsOpen(linkPairs(counter, 0), linkPairs(counter, 2)) = False
            linkSet(currentCount, linkCount) = Str(linkPairs(counter, 0)) & "." &
functions(functionVector(linkPairs(counter, 0))) & _
               ":Source:" & flavor
            If recurseLevel > 0 Then
               lastTap(recurseLevel) = lastTap(recurseLevel - 1) '- don't count sources as
moves?
            End If
            counter = counter + 1
            linkCount = linkCount + 1
         Else
            counter = counter + 1
         End If
      ElseIf linkPairs(counter, 1) = -1 Then
         ' protection buffer
```

```
        counter = counter + 1
      ElseIf inPortsOpen(linkPairs(counter, 0), linkPairs(counter, 2)) = True And outPortsOpen(linkPairs(counter, 1), linkPairs(counter, 2)) = True Then
        If linkPairs(counter, 2) = 0 Then
          flavor = "chocolate"
        ElseIf linkPairs(counter, 2) = 1 Then
          flavor = "vanilla"
        ElseIf linkPairs(counter, 2) = 2 Then
          flavor = "strawberry"
        End If
        linkSet(currentCount, linkCount) = Str(linkPairs(counter, 0)) & "." & functions(functionVector(linkPairs(counter, 0))) & _
          ":" & Str(linkPairs(counter, 1)) & "." & functions(functionVector(linkPairs(counter, 1))) & ":" & flavor
        lastTap(recurseLevel) = linkCount
        'inPortsOpen(linkPairs(counter, 0), linkPairs(counter, 2)) = False
        'outPortsOpen(linkPairs(counter, 1), linkPairs(counter, 2)) = False
        linkCount = linkCount + 1
        countUp = 0
        countUp2 = 0
        Do While inPortsOpen(countUp, 0) <> ""
          passIn(countUp, 0) = inPortsOpen(countUp, 0)
          passIn(countUp, 1) = inPortsOpen(countUp, 1)
          passIn(countUp, 2) = inPortsOpen(countUp, 2)
          passOut(countUp, 0) = outPortsOpen(countUp, 0)
          passOut(countUp, 1) = outPortsOpen(countUp, 1)
          passOut(countUp, 2) = outPortsOpen(countUp, 2)
          countUp = countUp + 1
        Loop
        passIn(linkPairs(counter, 0), linkPairs(counter, 2)) = False
        passOut(linkPairs(counter, 1), linkPairs(counter, 2)) = False
        counter = counter + 1
        stepDirection = linkerRecursion(functionVector, linkSet, passIn, passOut, length, functions, currentCount, linkCount, linkPairs, counter, recurseLevel + 1, lastTap)
        If stepDirection <> "cleared" Then
          For y = 0 To linkCount - 1
            linkSet(currentCount, y) = linkSet(currentCount - 1, y)
          Next
        End If
        If stepDirection = "return" Or stepDirection = "cleared" Then
          'lock down the appropriate parts of the in/out connections
          y = 0
          Do While linkSet(currentCount, y) <> ""
            splitA = Split(linkSet(currentCount, y), ":")
            flavorString = splitA(2)
            If flavorString = "chocolate" Then
```

```
            flavor = 0
          ElseIf flavorString = "vanilla" Then
            flavor = 1
          ElseIf flavorString = "strawberry" Then
            flavor = 2
          End If
          number1 = Split(splitA(0), ".")
          number1 = number1(0)
          number2 = Split(splitA(1), ".")
          If IsNumeric(number2(0)) Then
            number2 = number2(0)
            outPortsOpen(number2, flavor) = False
          Else
            ' do nothing
          End If
          inPortsOpen(number1, flavor) = False
          y = y + 1
        Loop
      End If
    Else
      counter = counter + 1
    End If
Loop
For n = 0 To 2
    For m = 0 To length - 1
      If inPortsOpen(m, n) = True Then
        countUp2 = countUp2 + 1
      End If
      If outPortsOpen(m, n) = True Then
        countUp = countUp + 1
      End If
    Next
Next

countUp = countUp + countUp2
If countUp > 1 And recurseLevel = 0 Then
    y = 0
    Do While linkSet(currentCount, y) <> ""
      linkSet(currentCount, y) = ""
      y = y + 1
    Loop
    linkerRecursion = "cleared"
ElseIf countUp > 1 And recurseLevel > 0 Then
    linkerRecursion = "cleared"
    y = lastTap(recurseLevel - 1)
    Do While y > -2 And linkSet(currentCount, y) > ""
```

```
            linkSet(currentCount, y) = ""
            linkCount = linkCount - 1
            y = y + 1
        Loop
    ElseIf stepDirection = "advance" Then
        currentCount = currentCount + 1
        stepDirection = "return"
        linkCount = linkCount - 1
        linkerRecursion = "return"
    End If
    If recurseLevel > 0 And linkerRecursion = "" Then
        linkerRecursion = "return"
    End If

End Function
```

APPENDIX C: S-CURVE SOLVING MATLAB CODE

This appendix contains a code listing for MATLAB to quickly fit a multidimensional S-curve to data. The code will also find the asymptotes of the S-curve that provide the best fit for data. The asymptote values can be constrained in order to work with the user's knowledge of physical limits and to prevent impossible situations, such as data points existing beyond the limits.

```
 dataset = 1;
% for martinoFighters
if dataset == 1
   no_vars = 4;
   X = zeros(8, 1);
   i = 1;
   L = [3.9; 5; 7500; 40];  % need guesses here
   y0 = [0.5; 0.2; 240; 0.0001]; % need guesses here
   X0 = [L; y0];
   lb = [2.5; 4.1; 5100; 31; 0.01; 0.05; 1; 0.0001; -Inf; -Inf];
   ub = [Inf; Inf; Inf; Inf; 0.55; 0.45; 350; 0.05; Inf; Inf];
   options = optimset('MaxFunEvals', 10000);
   [X(:,i), fval(i)] = fmincon(@multidimSfighter, X0, [], [], [], [], lb, ub, [], options);
   [fmval, timeVec, timeEstVec, b] = multidimSfighterVerify(X);
end
% for mooresLaw
if dataset == 2
   no_vars = 6;
   X = zeros(12, 1);
   i = 1;
   L = [0.1; 3.5; 60; 0.5; 50; 75];  % need guesses here
   y0 = [1; 95; 625; 30; 1; 1]; % need guesses here
   X0 = [L; y0];
   lb = [0.001; 1.5; 20; 0.1; 45; 70; 0.85; 90; 600; 30; 0.001; 0.001];
   ub = [0.14; 4; 65; 0.75; 150; 150; 1.5; 200; 1000; 75; 1.4; 1.6];
   options = optimset('MaxFunEvals', 10000);
   [X(:,i), fval(i)] = fmincon(@multidimMoores, X0, [], [], [], [], lb, ub, [], options);
   [fmval, timeVec, timeEstVec, b] = multidimMooresVerify(X);
end
% for dannerEngines
if dataset == 3
   no_vars = 3;
   X = zeros(6, 1);
```

```
    i = 1;
    L = [0.55; 7; 64000];  % need guesses here
    y0 = [0.75; 4; 17000]; % need guesses here
    X0 = [L; y0];
    lb = [0.05; 6.5; 61000; 0.67; 2; 10000];
    ub = [0.56; 30; 150000; 1.5; 4.5; 19000];
    options = optimset('MaxFunEvals', 10000);
    [X(:,i), fval(i)] = fmincon(@multidimEngines, X0, [], [], [], [], lb, ub, [], options);
    [fmval, timeVec, timeEstVec, b] = multidimEnginesVerify(X);
end

function fitness = multidimSfighter(invector)
% this is the place where a multi-dimensional S-curve fit to some data is
% evaluated
no_vars = floor(length(invector(:,1)) / 2);
L = invector(1:no_vars);
y0 = invector(no_vars + 1:no_vars*2);
datatable = [
0.65    1   500 0.1
0.8 1   366 0.1
0.9 1   480 0.1
0.75    0.5 500 5
0.7 0.5 500 0.1
1.4 0.5 500 0.1
1.6 0.5 1000    6
1.2 0.5 1000    6
2   0.5 1000    6
1.8 0.7 1000    0.1
2   0.6 1000    6
1.4 2.5 706 0.1];
time = [
1944
1946
1947
1949
1950
1953
1954
1953
1954
1955
1956
1959
];

xformtable = zeros(size(datatable));
```

```
for i = 1:length(datatable(:,1))
   for j = 1:length(datatable(1,:))
      xformtable(i, j) = log((L(j) - datatable(i,j)) / (datatable(i,j) - y0(j)));
   end
end
terms = zeros(no_vars, 1);
fitness = 0;
% use linear least-squares fitting
b = regress(time, [ones(length(xformtable(:,1)),1), xformtable]);
for i = 1:length(datatable(:,1))
   time_est = b(1);
   for j = 1:length(datatable(1,:))
      time_est = b(j + 1) * xformtable(i, j) + time_est;
   end
   fitness = fitness + (time(i) - time_est)^2;
end
function [fitness, time, timeE, b] = multidimSfighterVerify(invector)
% this is the place where a multi-dimensional S-curve fit to some data is
% evaluated
no_vars = floor(length(invector(:,1)) / 2);
L = invector(1:no_vars);
y0 = invector(no_vars + 1:no_vars*2);
datatable = [
0.65    1   500 0.1
0.8 1   366 0.1
0.9 1   480 0.1
0.75    0.5 500 5
0.7 0.5 500 0.1
1.4 0.5 500 0.1
1.6 0.5 1000    6
1.2 0.5 1000    6
2   0.5 1000    6
1.8 0.7 1000    0.1
2   0.6 1000    6
1.4 2.5 706 0.1
2.2 1.1 1000    10
2   0.7 2000    30
1.6 3   5000    0.1
2.4 2   2000    20
1.8 2.4 3000    0.1
1.8 3.2 3000    20
2   4   3000    20 ];
time = [
1944
1946
1947
```

```
1949
1950
1953
1954
1953
1954
1955
1956
1959
1967
1971
1971
1972
1974
1978
1982
];
xformtable = zeros(size(datatable));
for i = 1:length(datatable(:,1))
   for j = 1:length(datatable(1,:))
      xformtable(i, j) = log((L(j) - datatable(i,j)) / (datatable(i,j) - y0(j)));
   end
end
terms = zeros(no_vars, 1);
fitness = 0;
% use linear least-squares fitting
 b = regress(time, [ones(length(xformtable(:,1)),1), xformtable]);
for i = 1:length(datatable(:,1))
   time_est = b(1);
   for j = 1:length(datatable(1,:))
      time_est = b(j + 1) * xformtable(i, j) + time_est;
   end
   fitness = fitness + (time(i) - time_est)^2;
   timeE(i) = time_est;
end
timeE = timeE';
```

APPENDIX D: COMPILED TEST CASE COMPONENT DATA

This appendix contains the raw data that were collected on remote control helicopters, batteries, and permanent magnets. These data were then used to develop the technological component models that were used in Sindri for the test case.

Table 13. Battery cell chemistries

Tech	Time	Whr/kg	W/kg
NiCad	1960	40	130
NiMH	1986	65	300
LiCo	1994	140	300
NiMH	2000	75	500
LiPoly	2000	200	400
LiFePO4	2007	100	1500

Table 14. Commercial electric motors for remote control helicopters

Motor	Maker	Power (W)	Size (mm)	Weight (g)	Kv	Type
Park 180 Brushless Outrunner	Eflite	30	20	8.5	2200	Outrunner
Park 250 Brushless Outrunner	Eflite	55	23	14	2200	Outrunner
Park 300 Brushless Outrunner	Eflite	85	28	24	1380	Outrunner
Park 370 Brushless Inrunner	Eflite	84	30	48	3600	Inrunner
Park 370 Brushless Inrunner	Eflite	74	30	48	5400	Inrunner
Park 370 Brushless Outrunner	Eflite	150	28	45	1360	Outrunner
Park 400 Brushless Inrunner	Eflite	200	40	68	4200	Inrunner
Park 400 Brushless	Eflite	110	30	56	960	Outrunner

Outrunner						
Park 450 Brushless Outrunner	Eflite	150	33	72	890	Outrunner
Park 480 Brushless Outrunner	Eflite	230	35	87	1020	Outrunner
Six-Series Brushless Motor	Eflite	325	36	80	2000	Inrunner
Power 10 Brushless Motor	Eflite	340	43	122	1100	Outrunner
Power 15 Brushless Motor	Eflite	380	50	152	950	Outrunner
Power 25 Brushless Motor	Eflite	500	54	190	870	Outrunner
Power 32 Brushless Motor	Eflite	700	50	200	770	Outrunner
Rimfire 35-48-850 Brushless Outrunner	Great Planes	660	48	170	850	Outrunner
Ammo 12-30-3850 Brushless Inrunner	Great Planes	25	30	16	3850	Inrunner
Ammo 20-30-4300 Brushless Inrunner	Great Planes	94	30	45	4300	Inrunner
Ammo 36-50-3300 Brushless Inrunner	Great Planes	830	50	243	3300	Inrunner
Ammo 24-45-2350 Brushless Inrunner	Great Planes	270	45	101	2350	Inrunner
Ammo 28-56-2300 Brushless	Great Planes	590	56	166	2300	Inrunner

Inrunner						
Ammo 28-35-2200 Brushless Inrunner	Great Planes	260	35	94	2200	Inrunner
Rimfire 35-30-950 Brushless Outrunner	Great Planes	300	35	71	950	Outrunner
Rimfire 35-36-1500 Brushless Outrunner	Great Planes	550	36	102	1500	Outrunner
Rimfire 28-26-1000 Brushless Outrunner	Great Planes	135	28	41	1000	Outrunner
Rimfire 28-30-1250 Brushless Outrunner	Great Planes	200	30	54	1250	Outrunner
Pager motor 10 mm	Toytronics	5	12	5		Outrunner
Mighty Midget 10/7/19T	Mighty Midget	4	10	7		Outrunner
Mighty Midget 10/3/32T	Mighty Midget	2.5	10	3		Outrunner

Table 15. Remote control helicopters

Helicopter Name	rotor dia (in)	body wt (lb)	motor & fuel wt (lb)	motor max. dim
	7	0.0625	0.03684625	0.5
Blade CX2 (coaxial)	13.6	0.5	0.1439	1.2
ARTTECH Typhoon	25.75	1.40625	0.36	1.57
PROTECH small nitro	31	2	0.48	2.3
	49	5.15	1.4	2.8
	53	6.2	1.85	3.3
	60.6	8.4	1.9	3.45

Table 16. Commercial glow fuel engines for remote control flyers

Engine Name	power (hp)	weight (g)	max. dim (mm)	fuel flow (oz/hr)
Cox Pee Wee 0.02	0.035	25	41	1.4
Cox Tee Dee 0.02	0.055	21	41	2
Cox Black Widow 0.049	0.08	64	50	2.6
OS 10LA 0.11 in3	0.27	112	58	7.3
BigMig Sport 0.061	0.31	52	50	8.45
BigMig 0.0745 in3	0.38	70	53	10.14
OS15LA 0.15 in3	0.41	138	62	10.6
OS25LA 0.25 in3	0.6	200	71	14.8
ASP 0.15A Stunt	0.7	130	67	16.6
OS25FX 0.25 in3	0.84	250	71	19.9
BigMig 0.25	0.85	240	75	20.29
OS35AX 0.36 in3	1.3	360	78	28.6
Norvel AX-40 0.40 in3	1.4	310	82	30
OS45AX 0.45 in3	1.65	480	84	36.5

Table 17. Permanent magnet materials

Magnet Type	Energy Product B*H (MGOe)
~1940 - ferrites	3
~1950s - Platinum cobalt	7
~1968 - first Sm-Co	18
~1980 - samarium cobalt	30
~1988 - Neodymium	55

REFERENCES

[1] In Anderson, P.W., Arrow, K.J., Pines, D., ed. *Economy as an Evolving Complex System* (Addison-Wesley, Santa Fe, NM, 1987).

[2] In Arthur, W.B., Durlauf, S.N., Lane, D.A., ed. *The Economy as an Evolving Complex System II* (Addison-Wesley, Santa Fe, NM, 1997).

[3] Modeling and Simulation in Manufacturing and Defense Acquisition: Pathways to Success. (Board on Manufacturing and Engineering Design, 2002).

[4] History of TRIZ and I-TRIZ. 2002).

[5] Roadmap for the Office of Space Science Origins Theme. (National Aeronautics and Space Administration, 2003).

[6] CREAX Innovation Suite. (CREAX, 2003).

[7] JCIDS: Action Officer's Course. (US Marine Corps, 2004).

[8] Goldfire Innovator. (InventionMachine, 2004).

[9] Interactive TRIZ Matrix and 40 Principles. (SolidCreativity, 2004).

[10] Operation of the Defense Acquisition System. (Department of Defense, 2004).

[11] DoD Business Transformation. (Defense Acquisition University, 2004).

[12] Integrated Defense Acquisition, Technology, & Logistics Life Cycle Management Framework. p. Poster (Defense Acquisition University, 2005).

[13] Heliophysics: The New Science of the Sun-Solar Connection - Recommended Roadmap for Science and Technology 2005-2035. (National Aeronautics and Space Administration, 2005).

[14] Responding to the Call: Aviation Plan for American Leadership. (National Institute of Aerospace, Hampton, VA, 2005).

[15] NASA's Exploration Systems Architecture Study. (NASA, Washington, DC, 2005).

[16] JCIDS Manual (CJCSM 3170.01B). (Department of Defense, 2005).

[17] TRIZ Contrasolve V 1.0. (IDEACore, 2005).

[18] Innovation WorkBench. (Ideation International, 2005).

[19] What Is Technology? (National Academy of Engineering, 2006).

[20] Technology. *Wikipedia, the Free Encyclopedia* (Wikipedia, 2006).

[21] NIST/SEMATECH e-Handbook of Statistical Methods. (NIST, 2006).

[22] Software Architecture for Software-Intensive Systems. (Software Engineering Institute, 2006).

[23] DoDAF V1.5 Volume I: Definitions and Guidelines. (Department of Defense, 2007).

[24] Customers by Industry. (InventionMachine, 2008).

[25] CREAX:: Clients. (CREAX, 2008).

[26] About Us. (Ideation TRIZ, 2008).

[27] National Defense Budget Estimates for FY 2009. In Defense, D.o., ed (Office of the Undersecretary of Defense (Comptroller), Washington, DC, 2008).

[28] *JMP 8 Design of Experiments Guide*. (SAS Publishing, 2008).

[29] Picoo Z. *Wikipedia* (Wikipedia, 2009).

[30] A'Hearn, F.W., Bergman, C.E., Hirsch, E. Evolutionary Acquisition: An

Alternative Strategy for Acquiring Command and Control (C2) Systems. (DSMC Press, 1987).
[31] Aldridge, E.C. "Evolutionary Acquisition and Spiral Development." Memorandum. April 12 2002. (Department of Defense, 2002).
[32] Alsuwaiyel, M.H. *Algorithms: Design Techniques and Analysis*. (World Scientific Publishing Company, 1998).
[33] Anderson, P., Tushman, M.L. Technological Discontinuities and Dominant Designs: A Cyclical Model of Technological Change. *Administrative Science Quarterly*, 1990, **35**(4), 604-633.
[34] Arthur, W.B. The Logic of Invention. *Working Paper* (Santa Fe Institute, 2005).
[35] Arthur, W.B. The Structure of Invention. *Research Policy*, 2007, **36**(2), 274-287.
[36] Arthur, W.B., Polak, W. The Evolution of Technology within a Simple Computer Model. *Working Paper* (Santa Fe Institute, 2004).
[37] Ashdown, K. The RAH-66 Comanche: No Place in the Army of the Future. (Taxpayers for Common Sense, 2003).
[38] Baker, A.P. The Role of Mission Requirements, Vehicle Attributes, Technologies, and Uncertainty in Rotorcraft System Design. *Aerospace Engineering* (Georgia Institute of Technology, 2002).
[39] Basalla, G. *The Evolution of Technology*. (Cambridge University Press, 1989).
[40] Beinhocker, E.D. *The Origin of Wealth*. (Harvard Business School Press, Boston, MA, 2006).
[41] Beyer, H., Schwefel, H. Evolution strategies - A comprehensive introduction. *Natural Computing*, 2002, **1**(1), 3-52.
[42] Blackman, A.W., Seligman, E.J., Sogliero, G.C. An Innovation Index Based on Impact Analysis. *Technological Forecasting and Social Change*, 1973, **4**(3), 301-316.
[43] Blackmore, S.J. *The Meme Machine*. (Oxford University Press, Oxford, UK, 2000).
[44] Blickle, T., Teich, J., Thiele, L. Systems-Level Synthesis Using Evolutionary Algorithms. *Design Automation for Embedded Systems*, 1998, **3**, 23-58.
[45] Boehm, B. A spiral model of software development and enhancement. *Computer*, 1988, **21**(5), 61-72.
[46] Boehm, B. Sprial Development: Experience, Principles, and Refinements. (Carnegie Mellon University Software Engineering Institute, 2000).
[47] Bohorquez, F., Rankins, F., Baeder, J.D., Pines, D.J. Hover Performance of Rotor Blades at Low Reynolds Numbers for Rotary Wing Micro Air Vehicles. *21st Applied Aerodynamics Conference* (AIAA, Orlando, FL, 2003).
[48] Braha, D., Maimon, O. *A Mathematical Theory of Design: Foundations, Algorithms and Applications*. (Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998).
[49] Bray, O.H., Garcia, M.L. Fundamentals of Technology Roadmapping. (Sandia National Laboratories, 1997).
[50] Buonanno, M.A. A Method for Aircraft Concept Exploration using Multicriteria Interactive Genetic Algorithms. *Aerospace Engineering* (Georgia Institute of Technology, 2005).
[51] Callaway, D.S., Newman, M.E.J., Strogatz, S.H., Watts, D.J. Network Robustness and Fragility: Precolation on Random Graphs. *Physical Review Letters*, 2000, **85**, 5468-

5471.
[52] Campbell, P. *Permanent Magnet Materials and Their Application*. (Cambridge University Press, Cambridge, England, 1996).
[53] Chandler, D. Countdown for Rocket Planes. *Technology Review*2003).
[54] Colby, R.W. *The Encyclopedia Of Technical Market Indicators*. (McGraw-Hill, New York, NY, 2002).
[55] Constant, E.W. *The Origins of the Turbojet Revolution*. (John Hopkins University Press, Baltimore, MD, 1980).
[56] D'Aveni, R.A. *Hypercompetition*. (Free Press, 1994).
[57] Danner, T.W. A Formulation of Multidimensional Growth Models for the Assessment and Forecast of Technology Attributes. *Aerospace Engineering* (Georgia Institute of Technology, 2006).
[58] Dawkins, R. *The Extended Phenotype*. (Oxford University Press, Oxford, UK, 1982).
[59] Dawkins, R. *Climbing Mount Improbable*. (Norton, New York, NY, 1996).
[60] Dennett, D.C. *Darwin's Dangerous Idea*. (Simon & Schuster, New York, NY, 1995).
[61] Diestel, R. *Graph Theory*. (Birkhaeuser, Berlin, Germany, 2006).
[62] Dieter, G.E. *Engineering Design: A Materials and Processing Approach*. (McGraw-Hill, New York, NY, 2000).
[63] Dobzhansky, T. Nothing in Biology Makes Sense Except in the Light of Evolution. *American Biology Teacher*, 1973, **35**, 125-129.
[64] Dodson, E.N. Measurement of the State of the Art and Technological Advance. *Technological Forecasting and Social Change*, 1985, **27**(2), 129-146.
[65] Domb, E. Contradictions: Air Bag Applications. *TRIZ Journal*, 1997.
[66] Domb, E., Tate, K. 40 Inventive Principles. *TRIZ Journal*, 1997, **July 1997**.
[67] Dorigo, M., Di Caro, G. Ant colony optimization: a new meta-heuristic. *Proceedings of the Congress on Evolutionary Computation* (IEEE, 1999).
[68] Dosi, G. Technological paradigms and technological trajectories. *Research Policy*, 1982, **11**(3), 147-162.
[69] Emmerich, M., Groetzner, M., Schuetz, M. Design of Graph-Based Evolutionary Algorithms: A Case Study for Chemical Process Networks. *Evolutionary Computation*, 2001, **9**(3), 329-354.
[70] Engler, W.O., Biltgen, P.T., Mavris, D.N. Concept Selection Using an Interactive Reconfigurable Matrix of Alternatives (IRMA). *45th AIAA Aerospace Sciences Meeting and Exhibit* (AIAA, Reno, NV, 2007).
[71] Farkas, K., Thurston, P. Evolutionary Acquisition Strategies. (Air Force Institute of Technology, 2003).
[72] Fein, G. Current Operations Provide Lessons for Future Sea Basing Plans. *Defense Daily* (Defense Daily, 2008).
[73] Fey, V.R., Rivin, E.I. Guided Technology Evolution (TRIZ Technology Forecasting). *TRIZ Journal*, 1999, **January 1999**.
[74] Fisher, J.C., Pry, R.H. A Simple Substitution Model of Technological Change. *Technological Forecasting and Social Change*, 1971, **3**(1), 75-88.
[75] Fontana, W. Novelty in Evolution: Green Paper for Bio-Evolutionary Advanced Concepts for NASA. (Santa Fe Institute, 2001).

[76] Foster, R. When to Make Your Move to the Latest Innovation. *Across the Board*, 1986, **23**.
[77] Foster, R. *Innovation: The Attacker's Advantage*. (MacMillan, London, 1986).
[78] French, M.J. *Conceptual design for engineers*. (Springer, 1998).
[79] Fulghum, D.A. Unmanned Vehicles Mimic Insects. *Aviation Week and Space Technology* (Aviation Week Group, 2009).
[80] Gillett. PicoZ Tandem Z-1 and Micro MX-1 Extreme, Radio controlled toys. *The Guardian* (Guardian Media Group, London, UK, 2008).
[81] Globus, A., Atsatt, S., Lawton, J., Wipike, T. JavaGenes: Evolving Graphs with Crossover. (NASA, 2000).
[82] Gordon, T.J., Helmer, O. Report on a Long-Range Forecasting Study. (RAND Corporation, 1964).
[83] Gorn, M.H. *Prophecy Fulfilled: 'Toward New Horizons' and Its Legacy*. (Air Force History and Museums Program, 1994).
[84] Hadcock, R.N. The Cautious Course to Introducing New SDM Technology into Production Systems. *Astronautics and Aeronautics*, 1980(March 1980), 31-33.
[85] Hawthorne, S., Lush, R. Policies, News and Updates - Evolutionary Acquisition and Spiral Development. *CrossTalk*, 2002(August 2002).
[86] He, R., Sato, S., Drela, M. Design of a Single-Motor Nano Aerial Vehicle with a Gearless Torque-Cancelling Mechanism. *Aerospace Sciences Meeting and Exhibit* (AIAA, Reno, NV, 2008).
[87] Hirsch, E. JLC Guidance for Use of Evolutionary Acquisition to Acquire Weapons Systems. (DSMC Press, 1998).
[88] Holland, J.H. *Hidden Order: How Adaptation Builds Complexity*. (Helix Books, Reading, MA, 1995).
[89] Holland, J.H., Holyoak, K.J., Nisbett, R.E., Thagard, P.R. *Induction: Processes of Inference, Learning and Discovery*. (MIT Press, Cambridge, MA, 1986).
[90] Hollingsworth, P.M. Requirements Controlled Design: A Method of Discovery of Discontinuous System Boundaries in the Requirements Hyperspace. *Aerospace Engineering* (Georgia Institute of Technology, Atlanta, GA, 2004).
[91] Inman, O.L. Technology Forecasting using Data Envelopment Analysis. *Systems Science* (Portland State University, 2004).
[92] Inman, O.L., Anderson, T.R., Harmon, R.R. Predicting US jet fighter aircraft introductions from 1944 to 1982: A dogfight between regression and TFDEA. *Technological Forecasting and Social Change*, 2006, **73**(9), 1178-1187.
[93] Johnson, W. *Helicopter Theory*. (Dover Publications, New York, NY, 1994).
[94] Kane, J. A Primer for a New Cross-Impact Language - KSIM. *Technological Forecasting and Social Change*, 1972, **4**(2), 129-142.
[95] Kauffman, S.A., Levin, S. Towards a General Theory of Adaptive Walks on Rugged Landscapes. *Journal of Theoretically Biology*, 1987, **128**, 11-45.
[96] Kauffman, S.A., Lobo, J., Macready, W.G. Optimal search on a technology landscape. *Journal of Economic Behavior and Organization*, 2000, **43**, 141-166.
[97] Kawamura, Y., Souda, S., Nishimoto, S., Ellington, C.P. Clapping-wing Micro Air Vehicle of Insect Size. *Bio-mechanisms of Swimming and Flying* (Springer Japan, 2008).
[98] Kirby, M.R. A methodology for technology identification, evaluation, and

selection in conceptual and preliminary aircraft design. 2001).
[99] Kirby, M.R. An Approach for Strategic Planning of Future Technology Portfolios. (Georgia Institute of Technology Aerospace Systems Design Laboratory, 2006).
[100] Kirby, M.R., Mavris, D.N. A Method for Technology Selection Based on Benefit, Available Schedule and Budget Resources. *World Aviation Congress* (AIAA, San Diego, CA, 2000).
[101] Kirby, M.R., Mavris, D.N. An Approach for the Intelligent Assessment of Future Technology Portfolios. *40th AIAA Aerospace Sciences Meeting and Exhibit* (AIAA, Reno, NV, 2002).
[102] Kirby, M.R., Mavris, D.N., Largent, M.C. A Process for Tracking and Assessing Emerging Technology Development Programs for Resource Allocation. *1st AIAA Aircraft, Technology, Integration and Operations Forum* (AIAA, Los Angeles, CA, 2001).
[103] Klein, A. Weapons Updrage Faces Big Hurdles; Problems with Wireless Technology May Threaten Army's Ambitious Plan. *Washington Post* (Washington Post, Washington, DC, 2008).
[104] Kobryn, C., Sibbald, C. Modeling DoDAF Compliant Architectures. (Telelogic, 2004).
[105] Koza, J.R., Keane, M.A., Streeter, M.J. Evolving Inventions. *Scientific American* (Scientific American, New York, NY, 2003).
[106] Koza, J.R., Rice, J.P. *Genetic Programming*. (Springer, 1992).
[107] Lawler, E., Wood, D. Branch-and-bound methods: a survey. *Operations Research*, 1966, **14**(4), 699-719.
[108] Levinthal, D.A. The Slow Pace of Rapid Technological Change: Gradualism and Punctuation in Technological Change. *Industrial and Corporate Change*, 1998, **7**(2), 217-247.
[109] Loveridge, D., Georghiou, L., Nedeva, M. United Kingdom Technology Foresight Programme: Delphi Survey. (PREST, University of Manchester, 1995).
[110] Maier, M.W. *The Art of Systems Architecting*. (CRC Press, Boca Raton, FL, 2000).
[111] Mait, J. Micro Autonomous Systems and Technology (MAST) Collaborative Technology Alliance: Microsystems Vision (Presentation). (Army Research Laboratory, 2006).
[112] Mansfield, E. *The Economics of Technological Change*. (W.W. Norton and Company Inc., New York, 1968).
[113] Marchetti, C., Nakicenovic, N. The Dynamics of Energy Systems and the Logistic Substitution Model. (International Institute for Applied Systems Analysis, Laxenburg, Austria, 1979).
[114] Martino, J.P. A Comparison of Two Composite Measures of Technology. *Technological Forecasting and Social Change*, 1993, **44**(2), 147-159.
[115] Martino, J.P. *Technological Forecasting for Decision Making*. (McGraw-Hill, 1993).
[116] Martino, J.P. Thirty Years of Change and Stability. *Journal of Technology Forecasting and Social Change*, 1999, **62**, 13-18.
[117] Matticus78. Diagram of Eye Evolution. (Wikipedia, 2006).
[118] Mazur, G. The Theory of Inventive Problem Solving (TRIZ). 1995).

[119] McClure, E.K. An evolving-requirements technology assessment process for advanced propulsion concepts. *Aerospace Engineering* (Georgia Institute of Technology, 2006).
[120] Mokyr, J. Science, Technology and Knowledge: What Historians can Learn From an Evolutionary Approach. *Max Planck Institute on Evolutionary Economics Working Papers 9803*1998).
[121] Mokyr, J. Evolutionary Phenomena in Technological Change. In Ziman, J., ed. *Technological Innovation as an Evolutionary Process*, pp. 52-64 (Cambridge University Press, 1999).
[122] Mokyr, J. Natural History and Economic History: Is Technological Change an Evolutionary Process? 2000).
[123] Mosto, A. DoD Architecture Framework Overview. 2004).
[124] Nelson, R.R., Winter, S.G. *An Evolutionary Theory of Economic Change*. (Harvard University Press, Cambridge, MA, 1982).
[125] Padgett, J.F. The Emergence of Simple Ecologies of Skill. *The Economy as an Evolving Complex System II* (Addison-Wesley, Santa Fe, NM, 1998).
[126] Papadimitriou, C.H., Steiglitz, K. *Combinatorial Optimization: Algorithms and Complexity*. (Prentice-Hall, Upper Saddle River, NJ, 1998).
[127] Phelps, J. Dayton Contributes to the History of Magnetic Materials. (University of Dayton Research Institute, Dayton, OH, 1998).
[128] Raczynski, C.M. Technology Portfolio Assessments Using a Gene-Corrected Genetic Algorithm. *Aviation Technology, Integration, and Operations (ATIO)* (AIAA, Denver, CO, 2003).
[129] Raczynski, C.M. A methodology for comprehensive strategic planning and program prioritization. *Aerospace Engineering* (Georgia Institute of Technology, 2008).
[130] Radcliffe, N. Fitness variance of formae and performance prediction. *Foundations of Genetic Algorithms 3*, pp. 51-72 (Springer, Berlin, Germany, 1995).
[131] Radcliffe, N., Surry, P. Formal memetic algorithms. *Evolutionary Computation*, 1994, 1-16.
[132] Robel, I., Subramanian, V., Kuno, M., Karnat, P.V. Quantum Dot Solar Cells: Harvesting Light Energy with CdSe Nanocrystals Molecularly Linked to Mesoscopic TiO2 Films. *Journal of the American Chemical Society*, 2006, **128**(7), 2385-2394.
[133] Rosenbloom, R., Cusumano, M. Technological Pioneering and Competitive Advantage: The Birth of the VCR Industry. *California Management Review*, 1987, **29**(4), 51-67.
[134] Rosenbrock, H.H. An Automatic Method for Finding the Greatest or Least Value of a Function. *Computer Journal*, 1960, **3**, 175-184.
[135] Sakran, S.H., Koza, J.R., Jones, L.W. Automated Re-invention of a Previously Patented Optical Lens System Using Genetic Programming. *Genetic Programming - 8th European Conference* (Springer, Lausanna, Switzerland, 2005).
[136] Savransky, S.D. *Engineering of Creativity*. (CRC Press, Boca Raton, FL, 2000).
[137] Schields, K.J. Time-based acquisition programs: time-based and time-phased requirements within CJCSI 3170.01. *Air Force Journal of Logistics*, 2003(Winter 2003).
[138] Schittkowski, K. NLPQL: A FORTRAN subroutine solving constrained nonlinear programming problems. *Annals of Operations Research*, 1986, **5**(2), 485-500.
[139] Schumpeter, J.A. *The Theory of Economic Development*. 1911).

[140] Schwendiman, B. *Building Custom PHP Extensions*. (LULU, New York, NY, 2003).
[141] Seiden, A.e.a. P5 Report: The Particle Physics Roadmap. (Particle Physics Project Prioritization Panel, 2006).
[142] Snyder, J.F., Carter, R.H., Wong, E.L., et al. Multifunctional Structural Composite Batteries. (Army Research Laboratory, 2007).
[143] Sofge, E. Top 3 Robots Coming Soon to the Battlefield. *Popular Mechanics* (Hearst Corp., 2007).
[144] Souchkov, V. Four Views on TRIZ. *TRIZ Journal*, 1999, **March 1999**.
[145] Sweetman, W. Surveillance Options; Navy has multiple choices in meeting BAMS requirement. *Defense Technology International* (Aviation Week Group, 2008).
[146] Sylvester, R.K., Ferrara, J.A. Conflict and Ambiguity: Implementing Evolutionary Acquisition. *Acquisition Review Quarterly*, 2003(Winter 2003), 1-27.
[147] Vanderplaats, G.N. CONMIN: A FORTRAN program for the constrained function minimization: a User's manual. (NASA, Moffitt Field, CA, 1973).
[148] Villeneuve, F. A Method for Concept and Technology Exploration of Aerospace Architectures. *Aerospace Engineering* (Georgia Institute of Technology, Atlanta, GA, 2007).
[149] Villeneuve, F., Mavris, D.N. A New Method of Architecture Selection for Launch Vehicles. *AIAA/CIRA 13th International Space Planes and Hypersonics Systems and Technologies*Capua, Italy, 2005).
[150] Wasserman, S., Faust, K. *Social network analysis*. (Cambridge University Press, Cambridge, UK, 1994).
[151] Willyard, C.H., McClees, C.W. Motorola's Technology Roadmapping Process. *Research Management*, 1987, **30**(6), 13-20.
[152] Zlotin, B., Zusman, A. Directed Evolution - Philosophy, theory and practice. (Ideation International, Inc., 2001).
[153] Zlotin, B., Zusman, A. Patterns of Evolution: Recent Findings on Structure and Origin. 2006).
[154] Zwicky, F. *Discovery, Invention, Research - Through the Morphological Approach*. (Macmillan, Toronto, Canada, 1969).

VITA

Bjorn Cole was born in Olympia, Washington to Michael and Sandra Cole. He spent the majority of his formative years in the small town of Tenino, Washington and still has good friends in the area. After graduating high school in 1999, he went to Seattle to study aeronautics and astronautics at the University of Washington. He performed research in digital particle velocimetry with Dr. Dana Dabiri, and participated heavily in the student-run Mars Gravity Biosatellite project. In addition to another set of good friends, the academic atmosphere of the University introduced him to a variety of liberal arts, including political science, and allowed him to study theatre alongside engineering.

Upon graduation from Washington in 2003, Bjorn accepted a Graduate Research Assistanceship at the Aerospace Systems Design Laboratory at Georgia Tech. This provided him with opportunities with a variety of aerospace institutions, including Pratt and Whitney Rocketdyne, the Jet Propulsion Laboratory, NASA Langely, and multiple DoD research centers.

Bjorn's primary professional interest is the reinvigoration of the space industry, and expanding it to the point that it can support sustained basic research and human spaceflight through its own economy. He foresees manifold benefits from opening this realm of human experience, both materially and socially.

Bjorn looks forward to an exciting career in the space world, starting with a role at the Jet Propulsion Laboratory as a systems engineer.